An Evolutionary Approach to Maximum Likelihood Estimation for Generative Stochastic Models

Richard C Kelley [*] **Monica Nicolescu** [**] **Mircea Nicolescu** [***] **Sushil Louis** [****]

[*] *University of Nevada, Reno, Reno, NV 89557 USA. (email: rkelley@cse.unr.edu)*
[**] *e-mail: monica@cse.unr.edu*
[***] *e-mail: mircea@cse.unr.edu*
[****] *email: sushil@cse.unr.edu*

Abstract: Robotics problems essential for human-robot interaction and for the cognitive skills of robots often require or make use of maximum-likelihood estimation for hidden Markov models. Standard approaches to this problem rely on assumptions about the dependence structure of the variables in the model, and don't naturally generalize to parallel or distributed architectures. In this paper, we propose an evolutionary approach to address these issues. We show how a genetic algorithm can be used to perform maximum-likelihood estimation on a collection of jointly distributed random vectors, and illustrate our approach on the decoding problem for hidden Markov models. We provide experimental results showing the effectiveness of our method for specific problems in natural language processing and human-robot interaction. We also consider some of the limitations and possible extensions of this approach.

1. INTRODUCTION

Models of cognitive abilities in robots and other artificial systems often use probabilistic representations and inference algorithms to simulate human-like reasoning and behavior. The probabilistic approach has been highly successful (Thrun [2005]), as researchers have developed efficient algorithms for working with probabilistic systems. These algorithms often gain efficiency by exploiting assumptions about the probabilistic structure of problems, for example by analyzing conditional independence relations that are either assumed to exist for or are imposed on the variables in a model. The class of generative models, including the widely used hidden Markov model (HMMs), has proved to be an effective means for attacking problems related to cognitive processes and behaviors. For example, to model the way in which an agent's thoughts $\mathbf{X}$ are related to its actions $\mathbf{Y}$, we might use a model defined by a joint probability distribution such as $p(\mathbf{X}, \mathbf{Y})$, where $\mathbf{X}$ and $\mathbf{Y}$ are sequences satisfying the conditions that $\mathbf{X}$ is a Markov chain (which we assume cannot be directly observed) and $\mathbf{Y}_k$ is conditionally independent of all variables other than $\mathbf{X}_k$. This particular definition gives us the traditional hidden Markov model, though one could generalize this by altering the conditional independence assumptions of the model or by including additional random sequences in the joint distribution to model additional information available to the system, for example using a distribution such as $p(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$, along with some (possibly empty) set of conditions specifying the dependence relations among the three sequences.

Looking specifically at hidden Markov models, one sees that a large part of the reason for their effectiveness is the existence of an efficient dynamic-programming algorithm for solving the *decoding problem*: given a sequence of observations, determine a sequence of hidden states that "best explains" the observations. In this paper we discuss some of the limitations of the dynamic programming approach, and propose a genetic algorithm for solving the decoding problem. More generally, we describe how to use the evolutionary approach to perform maximum-likelihood estimation for a collection of jointly distributed random vectors, given known values for the members of an arbitrary subset of that collection. We also provide experimental results showing the effectiveness of our method for specific problems in natural language processing and human-robot interaction.

2. RELATED WORK

Hidden Markov models were initially developed by Baum and his colleagues in the late 1960s and early 1970s (Baum and Petrie [1966]). A standard reference on HMMs is the paper by Rabiner, which outlines the basic theory of HMMs as well as some results in one major application area, speech recognition (Rabiner [1989]). That paper also describes the decoding problem, as well as a dynamic programming algorithm to solve the decoding problem. That algorithm was first proposed by Viterbi (Viterbi [1967]). Although this algorithm gives the optimal sequence and is efficient, it does not naturally generalize to more complicated joint distributions, and even in the hidden Markov case is not suitable for all applications in which decoding may be necessary. Additionally, although parallel versions of the Viterbi algorithm have been developed for particular parallel architectures such as systolic arrays (Reeve and Amarasinghe [2006]), comparatively little research has been done on decoding algorithms that work for more general distributed architectures.

Genetic algorithms were pioneered by Holland (Holland [1992]) and have seen their use expand tremendously as computational power has increased. In the context of statistical inference, genetic algorithms have been successfully used to perform parameter estimation (Sharman and McClurkin [1989]), and specifically for HMMs, have been used to aid in the estimation of model parameters from sample data (Slimane et al. [1996]). The current work applies the idea of using genetic algorithms for inference problems to the general class of problems that can be expressed in terms of maximizing a joint distribution, and looks particularly at how this approach can be used in the case of hidden Markov models.

3. THE METHOD

3.1 Problem Statement

We consider two closely related problems, both solvable by our method. In the first problem, we consider a collection of random vectors, all of whose components are discrete random variables; in the second we consider just a collection of discrete-valued random variables. The second problem is clearly expressible in terms of the first, so that our solution to the vector version of the problem gives us a solution to the random variable version. However, depending on the particulars of the model being developed, one approach may be more natural to work with than the other. However, since the second problem is a special case of the first, the basic problem to solve can be described as maximum likelihood inference for a collection of random vectors where only some of those vectors have known values or can be observed.

Random Vector Version This version of the problem assumes that we have n random vectors X_i, $i = 1, \ldots, n$, each of whose components are discrete random variables. The component variables may range over different sets, and we make no assumptions about the conditional independence relationships among the component variables or the vectors (though we show below how this information can be used if it is available). Given an assignment to some collection of the vectors, the problem is to find assignments to the other vectors that maximize the joint probability, $p(X_1, \ldots, X_n)$.

When we have the conditions

(1) $n = 2$,
(2) X_1 is not directly observable (so we are never explicitly given values for the components of X_1),
(3) the dependence structure of X_1 satisfies the Markov property, and
(4) X_2's components are independent of everything except the corresponding components of X_1,

we obtain the common case of a hidden Markov model. In this case, the problem is equivalent to the *decoding* problem: given a vector of observations X_2, determine the vector X_1 that maximizes $p(X_1 \mid X_2)$. It turns out, as shown below, that this is equivalent to maximizing $p(X_1, X_2)$, so that decoding is an instance of the vector version of our maximum likelihood problem.

One final assumption we make is that we have, or can estimate, the joint distribution $p(X_1, \ldots, X_n)$ for any assignment of values to all n variables. We have found that in most practical applications this assumption can be met with little difficulty.

Random Variable Version For this version of the problem, let $X_1, \ldots, X_n$ be n discrete random variables, and consider the joint distribution of these random variables, $p(X_1, \ldots, X_n)$. In this case, we may formally state our problem as follows: given an assignment of values to some subset of the X_i's, say $X_{i_1} = x_{i_1}, \ldots, X_{i_k} = x_{i_k}$, find values of the remaining unassigned variables that maximize the joint probability $p(X_1, \ldots, X_n)$.

That this is a special case of the vector version of the problem may be easily seen by letting each vector have a single component corresponding to one of the random variables X_i in the second formulation of the problem. Because of this, we address only the vector version of the problem in the following discussion.

3.2 Solution

To solve the above problems, we propose the use of a genetic algorithm. A genetic algorithm consists of a "population" of possible solutions, a way of evaluating the quality of each of these candidates, and a set of operators to transform the population in ways that are intended to increase the average quality of the candidates in the population over time.

To specify a genetic algorithm, one must provide a representation for possible solutions to the problem being modeled, a fitness function whose value we wish to maximize, and the operators for generating a new population from the current population at each time step. We consider each of these in turn.

Solution Representation and Population Size When using a genetic algorithm, one begins by specifying a collection of candidate solutions. In our case, we assume that we have n random vectors $X_1, \ldots, X_n$, and that some of these vectors have values assigned to them. We then seek the most likely assignment of values to the remaining vectors. Since a potential solution is given once we have values for all n vectors, we represent each candidate solution as a list of the components of each of the n vectors, in a fixed order. If each vector has k components, then a single member of the population will be a list of the kn values from the vectors, concatenated together in a list. The number of individuals in the population is a parameter that can vary depending on the problem, but in the experiments below we found that a population size consisting of as few as one to two hundred individuals sufficed to obtain acceptable results.

Fitness Function Given an assignment of values $x_1, \ldots, x_n$ to the random vectors $X_1, \ldots, X_n$, we use the fitness function $p(x_1, \ldots, x_n)$. Thus, more likely assignments to the vectors will have higher fitness, and one can expect that the genetic algorithm will, given sufficient resources, converge to an assignment of the vectors that has a high probability relative to the members of the initial population.

Operators We use standard genetic operators for our problem. In particular, we use operators for crossover, mutation, and selection. In our experiments, we consider several different approaches to generating a new population from the current population. These approaches include:

(1) a simple genetic algorithm with elitism,
(2) a steady-state genetic algorithm in which only a fraction of the population is replaced in each generation, and
(3) a multi-population "deme GA" that is naturally suited to implementation on a parallel computing architecture.

We discuss these variants further in our results section.

Recovering the Joint Distribution; Inference in More General Models In the course of modeling a process using random vectors $X_1, \dots, X_n$, it may be necessary or convenient to specify some kind of dependence structure on the random vectors or their components. In the most general case, we may not have direct access to the joint distribution $p(X_1, \dots, X_n)$, but only to some collection of conditional distributions of some of the X_i's in terms of the others. Perhaps the simplest example of this phenomenon is the discrete hidden Markov model, where instead of a joint distribution $p(\mathbf{X}, \mathbf{Y})$ to use for decoding, we are usually presented with tables describing the two conditional distributions over the components of $\mathbf{X}$ and $\mathbf{Y}$ (here indexed by k): $p(X_{k+1} \mid X_k)$ and $p(Y_k \mid X_k)$, with the X_k's taking values in a hidden state space and the Y_k's taking values among a set of visible symbols. In certain cases, it may be possible to use the given conditional distributions for a general model to recover the joint distribution $p(X_1, \dots, X_n)$ for any particular assignment of values to the X_i's. Where this is possible, the evolutionary approach outlined above can be used to perform maximum-likelihood inference for a subset of the X_i's given an assignment of values to the others.

The approach to a problem then becomes to recover the joint distribution from the conditional distributions: we first compute the joint distribution in terms of the conditional distributions that are available, and then using the computed joint distribution, proceed as outlined above, using a genetic algorithm to find the most likely assignment of values to all the vectors in the model. Here the applicability of the method hinges entirely on the ease with which the designer can compute or approximate $p(X_1, \dots, X_n)$ for an arbitrary assignment of values to the X_i's.

4. EXPERIMENTAL RESULTS

4.1 The Hidden Markov Model Case

As mentioned above, for models with very general dependence structures, it may be difficult to calculate the required joint probabilities in a reasonable amount of time. However, in some cases it is quite easy; one case where such a calculation can be performed is with HMMs. The basic idea (as outlined above) is to use the conditional distributions that determine an HMM to recover the joint distribution for use as the fitness function of the genetic algorithm. We now outline the derivation showing that this is possible for hidden Markov models.

Following Rabiner [1989], we denote our vector of T observations by $O = O_1 O_2 \dots O_T$. If Q is a vector of hidden states, we write $Q = q_1 \dots q_T$ for the individual hidden states of Q. We also let π_{q_i} denote the probability that the first hidden state is q_i, we let $a_{q_i q_j}$ denote the probability of transitioning from state q_i to state q_j, and we let $b_{q_i}(O_i)$ denote the probability of observing visible symbol O_i while in state q_i.

The decoding problem asks us to find

$$\arg\max_Q P(Q \mid O).$$

By Bayes's rule, we have

$$P(Q \mid O) = \frac{P(O \mid Q) P(Q)}{P(O)}.$$

Since $P(O)$ is independent of our choice of Q, we can ignore it without affecting the solution to our problem. Thus the decoding problem is equivalent to

$$\arg\max_Q P(O \mid Q) P(Q),$$

which by the definition of conditional probability is identical to

$$\arg\max_Q P(O, Q),$$

showing that decoding is a special case of the vector maximum likelihood problem above. Moreover, since the logarithm is a strictly increasing function, we can solve the equivalent problem

$$\arg\max_Q \log P(O, Q),$$

often simplifying the computation. This shows that decoding is an instance of the maximum likelihood problem; we next show how we can efficiently compute this probability in terms of the parameters that define an HMM.

Although we are maximizing $\log P(O, Q)$, we need to express it in terms of the distributions that are used to define the HMM, namely the $a_{q_i q_j}$'s and the $b_{q_i}(O_i)$'s. In particular, fitness function that we use is

$$\begin{aligned}
\log P(Q, O) &= \log \left(P(O \mid Q) P(Q) \right) \\
&= \log P(O \mid Q) + \log P(Q) \\
&= \log \left(\prod_{t=1}^{T} b_{q_t}(O_t) \right) + \log \left(\pi_{q_1} \prod_{t=1}^{T-1} a_{q_t q_{t+1}} \right) \\
&= \log \pi_{q_1} + \log b_{q_T}(O_T) \\
&\quad + \sum_{t=1}^{T-1} \left[\log b_{q_t}(O_t) + \log a_{q_t q_{t+1}} \right].
\end{aligned}$$

This is an instance of recovering the joint distribution from a set of conditional distributions; if we store the logarithms of the probabilities of the HMM, then the fitness function can be computed using only addition, with only an asymptotically linear number of arithmetic operations. By scaling the log-probabilities appropriately, this objective function allows us to work with very long sequences of observations that could otherwise present numerical difficulties for the algorithm.

4.2 Experimental Setup and Results

To verify our approach in the HMM case, we test it on three different models. First, we consider an artificial two-state chain that comes from no particular application area, and then we examine the effectiveness of the evolutionary decoding method for part-of-speech tagging in natural language processing and intent recognition in robotics. But first we introduce the particular genetic algorithms we tested and the metric we use to evaluate our approach.

Genetic Algorithms To test the method, we use three different versions of the genetic algorithm. The first is a simple genetic algorithm, as described in Goldberg [1989]. This algorithm replaces the entire population for each generation. The second algorithm we consider is a steady-state genetic algorithm, in which only a fraction of the population is replaced. The replacement fraction p is a parameter that we must set, and for these experiments we use $p = 0.1$, $p = 0.25$, and $p = 0.5$. Lastly, we use a deme-based genetic algorithm. This GA consists of multiple populations with migration between the populations. At the end of each generation, a fraction of the best individuals from one population replace the worst individuals in a neighboring population. Both the migration percentage m and the number of populations k are parameters that we control. For these experiments, we consider $m = 0.2$, $m = 0.4$, $k = 2$, $k = 4$, $k = 8$, and $k = 16$.

Measuring Against Viterbi To measure the quality of the output of our algorithm, for each string we decode in the models below, we compare the results of decoding with our method to the results of decoding using the Viterbi algorithm. In particular, if we have n sequences of observations to decode, then since the Viterbi algorithm is optimal (Rabiner [1989]), it will produce the optimal state sequence for each of the n observation sequences. Let n_{ev} be the number of sequences for which evolutionary decoding produces a maximally likely hidden state sequence. Then we use

$$\gamma = \frac{n_{ev}}{n},$$

the fraction of correctly decoded sequences, as a measure of the quality of the evolutionary method. We then hope that in our applications γ is close to 1; we'll see below for our artificial model and two application models that this indeed is the case.

Decoding A Simple Model We start with a simple artificial model. The model consists of two hidden states, labeled 0 and 1, and six observable symbols, a, b, c, d, e, and f. The transition and emission probabilities were chosen by picking random numbers for each transition and normalizing to produce probabilities. The probabilities themselves are listed below in Table 1 and in Table 2. Given these probabilities, we generated one thousand random sequences of observations of varying lengths, and then decoded each sequence. Table 3 shows the γ values for these $n = 1000$ sequences, for each of the three types of genetic algorithm we tested.

	0	1
0	0.154	0.846
1	0.271	.729

Table 1. State transition probabilities for the simple model.

	a	b	c	d	e	f
0	0.355	0.291	0.022	0.332	0.0	0.0
1	0.169	0.272	0.0	0.0	0.283	0.276

Table 2. Emission probabilities for the simple model.

Genetic Algorithm Used	γ
Simple GA	0.98
Steady-State GA ($p = 0.10$)	0.98
Steady-State GA ($p = 0.25$)	0.985
Steady-State GA ($p = 0.5$)	0.986
Deme GA ($m = 0.2$ and $k = 2$)	0.971
Deme GA ($m = 0.2$ and $k = 4$)	0.976
Deme GA ($m = 0.2$ and $k = 8$)	0.977
Deme GA ($m = 0.2$ and $k = 16$)	0.978
Deme GA ($m = 0.4$ and $k = 2$)	0.969
Deme GA ($m = 0.4$ and $k = 4$)	0.978
Deme GA ($m = 0.4$ and $k = 8$)	0.98
Deme GA ($m = 0.4$ and $k = 16$)	0.982

Table 3. γ values for the simple HMM.

The results above allow us to make a few interesting comparisons. First, the simple algorithm works fairly well in comparison to the optimal Viterbi algorithm. In many applications, getting the optimal answer 98% of the time may be perfectly acceptable. We can also say a bit about how varying the parameters for the steady-state and deme algorithms affects performance. The steady-state version seems to perform better on this problem when more of the population is replaced in each generation; however, the steady-state GA with $p = 0.10$ tends to be more efficient than the other steady-state alternatives, and the difference in quality between strategies that use little replacement and those that use more replacement is small enough that the extra speed that comes from using a small replacement percentage may more than offset the loss of accuracy in some applications.

We can perform a similar analysis varying the parameters of the deme GA. Generally speaking, the performance of the deme algorithm improves (as one would expect) as we increase the number of populations and the fraction of each population that migrates to a new population in each generation.

Comparing the three different algorithms to each other, we see that the steady-state algorithm outperforms the simple algorithm, and that when few populations are used in the deme GA, its performance is poor compared to the other algorithms. However, as we increase the number of populations used for the deme GA from 2 to 16, we see γ increase to the point that it is comparable to the other algorithms in quality.

Part-of-Speech Tagging Our first non-trivial HMM comes from the field of computational linguistics, and is based on one approach to the *part-of-speech tagging* problem. A solution to this problem is an algorithm that takes as input a phrase or sentence in a natural language such as English, and produces a sequence of "tags," such that the ith tag represents the part of speech of the ith word in the input sentence. For instance, the sentence

Snow is cold.

Should receive the tag sequence

NN VBZ JJ

Where the symbol "NN" denotes a singular or mass noun, the symbol "VBZ" represents a 3rd person singular present verb, "JJ" represents an adjective. Hidden Markov models and their variants play a large role in part-of-speech tagging, which in turn is often used in applications such as partial parsing, lexical acquisition, information extraction, and question answering (Manning and Schütze [1999]).

The hidden Markov model we use was developed by Michael Collins for his PhD dissertation (Collins [1999]). Its hidden states are the possible parts of speech for English, and the visible symbols are words in the English language. The model consists of 44 hidden states, the hidden-state transition table consists of 1,981 transition probabilities, and the emission probability table consists of 119,287 emission probabilities.

To test our method on this model, we used a collection of phrases based on English phrases that appear in the 1990 Part-of-Speech Tagging Guidelines for the Penn Treebank Project (Santorini [1990]). Examples of such phrases include:

Either child could sing.
Either a boy or a girl could sing.
One of the best reasons
The girls all left.
There was a party in progress there.

Again we use the γ-value as our metric for evaluating the evolutionary approach. The results for the three different algorithms are shown in Table 4. The results show again

Genetic Algorithm Used	γ
Simple GA	0.92
Steady-State GA ($p = 0.10$)	0.98
Steady-State GA ($p = 0.25$)	0.94
Steady-State GA ($p = 0.5$)	0.96
Deme GA ($m = 0.2$ and $k = 2$)	0.86
Deme GA ($m = 0.2$ and $k = 4$)	0.96
Deme GA ($m = 0.2$ and $k = 8$)	0.96
Deme GA ($m = 0.2$ and $k = 16$)	0.98
Deme GA ($m = 0.4$ and $k = 2$)	0.90
Deme GA ($m = 0.4$ and $k = 4$)	0.96
Deme GA ($m = 0.4$ and $k = 8$)	0.98
Deme GA ($m = 0.4$ and $k = 16$)	0.96

Table 4. γ values for the part-of-speech HMM.

that the method works fairly well. The lessons here are similar to those from the simple artificial model. The simple GA's performance is slightly worse in this case, but the steady-state GA continues to perform very well, especially with a small replacement percentage. The deme GA performs poorly for a small number of populations, and then becomes competitive with the other algorithms as we increase the number of populations used.

Intent Recognition Tavakkoli et al. [2007] develop an approach to intent recognition using hidden Markov models. The *intent recognition problem* is an important problem in social robotics and computer vision. As defined in Tavakkoli et al. [2007], given a video sequence of one or more agents performing various activities, assign to each agent in each frame a label corresponding to that agent's current activity. In addition, determine in each frame the intentional (goal-directed) mental states of each agent. To use a hidden Markov model to solve this problem, one models the mental states of the agent as the hidden states of the HMM, and models the visible changes in the agent's activities using the visible variables of the model.

In Tavakkoli et al. [2007], one of the models considered is called *following*. This model represents the intentional states of an agent that is in the process of following another agent. The model was trained by having a robot follow a human, recording its observations as it performed the task. The collected observation sequences were then used as input to the Baum-Welch algorithm to train the model and obtain the parameters in the state-transition table and visible symbol emission table.

To test the evolutionary decoding algorithm on this model, we use observations collected by a mobile robot observing one human following another. Specifically, we use 30 sequences corresponding to disjoint segments of six video clips captured by the robot. Each video sequence consisted of one person following another person, with the robot observing the scene from several different perspectives.

We use the same three genetic algorithms for this problem that we do for the others. The γ values for each are listed in Table 5 The results show that for this problem the

Genetic Algorithm Used	γ
Simple GA	1.0
Steady-State GA ($p = 0.10$)	1.0
Steady-State GA ($p = 0.25$)	1.0
Steady-State GA ($p = 0.5$)	1.0
Deme GA ($m = 0.2$ and $k = 2$)	0.63
Deme GA ($m = 0.2$ and $k = 4$)	0.87
Deme GA ($m = 0.2$ and $k = 8$)	1.0
Deme GA ($m = 0.2$ and $k = 16$)	1.0
Deme GA ($m = 0.4$ and $k = 2$)	0.53
Deme GA ($m = 0.4$ and $k = 4$)	0.93
Deme GA ($m = 0.4$ and $k = 8$)	0.97
Deme GA ($m = 0.4$ and $k = 16$)	1.0

Table 5. γ values for the follow HMM.

simple and steady-state GAs perform exceptionally well, correctly decoding all of the input sequences given. The deme GA again performs poorly for a small number of populations, giving much worse results for this problem than the other two we consider. However, we again see the quality of the algorithm increase as the number of populations increases. This seems to be a general feature of the proposed algorithm.

5. DISCUSSION

While the numerical results of the previous section show some of the promise of the GA-based approach to the family of problems we consider, there are several other points worth making about the method that we propose.

5.1 Limitations

Although the evolutionary algorithm compares well to the Viterbi algorithm for decoding hidden Markov models, it does have limitations. The most important of these is the time necessary for the algorithm to converge on an optimal solution. For a large population of long vectors (consisting of hundreds of components), the number of generations required to find a decent solution to the problem may be large. This means that in some cases (even some listed above), the amount of time required for the genetic algorithm to find a near-optimal solution may be prohibitive. In the next subsection, though, we outline one benefit of the evolutionary approach that has the potential to mitigate this problem substantially.

5.2 Parallelism and Resource Adaptiveness

From the descriptions above, one can see that the deme-based genetic algorithm maps naturally to a parallel processing environment. This advantage will become increasingly decisive as multiprocessors and thread-level parallelism become more ubiquitous throughout computing (Hennessy and Patterson [2007]). It is only reasonable to expect that these trends will greatly impact computing in robotics as well. One of the advantages of the evolutionary approach to decoding (and maximum-likelihood more generally) is that it reduces the problem of developing a parallel decoding or maximum likelihood algorithm (a less well-studied problem) to the problem of building a parallel genetic algorithm (a *very* well-studied problem). Indeed one of the strongest attractions of genetic algorithms is that they are almost trivially parallelizable, with parallel algorithms that tend to perform well (Gordon [1993]), allowing us to easily exploit increasing parallelism in hardware. One would expect (though this should be verified experimentally) that as parallel computing resources grow, the evolutionary maximum-likelihood method should scale easily to deal with very large estimation problems.

6. CONCLUSION AND FUTURE WORK

In this paper, we proposed a new approach to maximum likelihood estimation for problems involving jointly-distributed random vectors. This approach relies on a genetic algorithm to find an assignment of values that maximizes the joint probability of the variables. We showed how the common problem of decoding a hidden Markov model can be viewed as an instance of our vector maximum likelihood problem, and provided experimental results for this case. We also discussed the possible efficiency problems for the genetic algorithms, and indicated how this can be dealt with using parallism, which is a natural and well-studied technique in the evolutionary computing literature.

We are currently exploring several possible extensions to this work. First, we are working on developing more general stochastic models that can use our evolutionary maximum likelihood approach to solve practical problems analogous to decoding a hidden Markov model. Secondly, we are looking at developing an efficient *online* version of the algorithm for robotics applications with hard real-time constraints. Lastly, we are researching methods for extending the evolutionary approach to other statistical problems related to robot cognition.

ACKNOWLEDGEMENTS

We would like to thank Catherine Porter for helpful discussions about our approach and for reviewing an early draft of this paper. The software for this work used the GAlib genetic algorithm package, written by Matthew Wall at the Massachusetts Institute of Technology. This work has been supported by the Office of Naval Research under award N00014-06-1-0611.

REFERENCES

S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, MIT Press, 2005.

L.E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state Markov chains", Ann. Math. Stat., vol. 37, pp. 1554-1563, 1966.

A.J. Viterbi, "Error bounds for convolution codes and an asymptotically optimal decoding algorithm", IEEE Trans. Informat. Theory, vol. IT-13, pp. 260-269, Apr. 1967.

L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition", Proceedings of the IEEE, 77(2), Feb. 1989.

J.S. Reeve and K. Amarasinghe, "A parallel Viterbi decoder for block cyclic and convolution codes", Signal Process. 86, 2, Feb. 2006.

J. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, 1992.

K.C. Sharman and G.D. McClurkin, "Genetic algorithms for maximum likelihood parameter estimation", International Conference on Acoustics, Speech, and Signal Processing, 1989, vol. 4, pp. 2716 - 2719, May 1989.

D. Goldberg *Genetic Algorithms in Search, Optimization, and Machine Learning* Addison-Wesley, 1989.

M. Slimane, G. Venturini, J. Beauville, T. Brouard, A. Brandeau, "Optimizing hidden Markov models with a genetic algorithm", AE '95: Selected Papers from the European conference on Artificial Evolution, pp. 384-396, Springer-Verlag, 1996.

C. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*, MIT Press, 1999.

M. Collins, "Head-driven statistical models for natural language parsing", PhD Dissertation, University of Pennsylvania, 1999.

B. Santorini, "Part-of-speech tagging guidelines for the Penn Treebank Project", 3rd revision, 2nd printing, June 1990.

A. Tavakkoli, R. Kelley, C. King, M. Nicolescu, M. Nicolescu, G. Bebis, "A vision-based architecture for intent recognition", Proceedings of the 3d International Symposium on Visual Computing, November 2007.

J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kauffman, 2007.

V.S. Gordon and D. Whitley, "Serial and parallel algorithms as function optimizers", ICGA-93, pp. 177-183, 1993.