



Gene recognition based on DAG shortest paths

John S. Chuang and Dan Roth

Department of Computer Science, University of Illinois at Urbana-Champaign, Digital Computing Laboratory, Urbana, Illinois, 61801, USA

ABSTRACT

We describe DAGGER, an *ab initio* gene recognition program which combines the output of high dimensional signal sensors in an intuitive gene model based on directed acyclic graphs. In the first stage, candidate start, donor, acceptor, and stop sites are scored using the SNoW learning architecture. These sites are then used to generate a directed acyclic graph in which each source-sink path represents a possible gene structure. Training sequences are used to optimize an edge weighting function so that the shortest source-sink path maximizes exon-level prediction accuracy. Experimental evaluation of prediction accuracy on two benchmark data sets demonstrates that DAGGER is competitive with *ab initio* gene finding programs based on Hidden Markov Models.

Contact: jsc@ocf.berkeley.edu

INTRODUCTION

Accurate and reliable computational gene structure prediction remains a challenge in large eukaryotic genomes. The necessary consideration of pre-mRNA splicing and the low coding sequence density complicates prediction by dramatically increasing the number of possible gene structures. This difficult combinatorial problem has fueled the development of a variety of gene finding programs (reviewed in Haussler, 1998; Burge and Karlin, 1998; Fickett, 1996) which can be coarsely classified as *ab initio* or sequence similarity-based. An example of the latter class is PROCUSTES (Gelfand et al., 1996), which implements a spliced alignment algorithm designed to explore different segmentations of a query genomic sequence, attempting to maximize amino acid sequence similarity to a target protein homolog. Sequence similarity-based programs can make very accurate gene predictions if a close homolog is known, but accuracy may decline significantly if more distant (BLASTX P-value $> 10^{-50}$) homologs are used (Guigo et al., 2000).

In contrast to homology-based programs, *ab initio* gene finders typically incorporate modules trained to recognize specific sequence features (transcriptional, splicing, translational) in an integrated gene model used for prediction. A variety of statistical and pattern recognition methods have been employed, and benchmark test sets have been developed to compare the accuracy of

existing programs (Buret and Guigo, 1996; Kulp et al., 1996; Reese et al., 1997). Due to their relatively high accuracy and an underlying gene model with an appealing probabilistic interpretation, gene finders based on hidden Markov models (HMMs; Rabiner, 1989) have received significant attention. Examples include VEIL (Henderson et al., 1997), Genie (Kulp et al., 1996; Reese et al., 1997), HMMgene (Krogh, 1997), and Genscan (Burge and Karlin, 1997). The benefit of incorporating homology information into *ab initio* gene finders has also been demonstrated (Buret and Guigo, 1996; Reese et al., 1997), leading to a hybrid class of programs.

Although HMM-based programs have dominated the gene finding landscape recently, we have taken a different approach. In this report, we describe a *ab initio* gene prediction program which makes use of a system of learned classifiers within a constraint satisfaction and optimization framework. Potential sites (starts, donors, acceptors, stops) are first scored using high dimensional classifiers based on a Sparse Network of Winnows (SNoW Roth, 1998). Subsequently, the problem is converted into a single source shortest path search on a directed acyclic graph (DAG) incorporating gene structure constraints. An edge weighting function is chosen to optimize prediction accuracy. Shortest paths in DAGs can be computed efficiently by a standard algorithm based on topological sorting, which is in turn based on depth first search (Cormen et al., 1990). Ranked, suboptimal paths can also be generated using *k*-shortest path algorithms (Azevedo et al., 1993; Martins and Santos, 2000). Other graph-theoretic optimization methods could be used within this algorithmic framework.

In principle, the approach can accommodate the incorporation of multiple sources of information into the constraints satisfaction and optimization stage. At this point, we only use a simple gene model; despite this, prediction accuracy on benchmark data sets already compares favorably with other *ab initio* methods and shows promise. The program will be referred to as DAGGER, short for Directed Acyclic Graph GENE Recognition.

The rest of this paper is organized as follows: In the next section, we provide a brief overview of SNoW and its application to site scoring. We then describe the procedure for graph construction, the selection of edge

weights, and the shortest paths-based prediction of gene structure. Finally, in the remaining sections, we present an evaluation of DAGGER’s accuracy in comparison to existing programs and outline some desirable avenues for future development.

SNOW-BASED SITE SCORING

Candidate start, donor, acceptor, and stop sites (each occurrence of ATG, GT, AG, and a stop codon, respectively) are scored using the SNoW learning system (Carlson et al., 1999; Roth, 1998)[†]. SNoW is a multi-class classifier that is based on Winnow multiplicative weight-update algorithm (Littlestone, 1988); it is tailored for learning in domains in which the potential number of features taking part in decisions is very large, but may be unknown a priori, by exploiting sparseness in the data. The system has been applied to learning problems in natural language processing involving high-dimensional feature spaces. Some examples include context-sensitive spelling correction (Golding and Roth, 1999), part-of-speech tagging (Roth and Zelenko, 1998), and identifying phrases in sentences (Munoz et al., 1999; Punyakanok and Roth, 2001) (which is somewhat analogous to identifying exons in DNA sequences).

The most basic aspects of the SNoW learning architecture are linear units referred to as *target nodes* and their sparse connection to a set of input layer features. Separate subnetworks, each containing two target nodes corresponding to *true* sites and *pseudo* sites, are employed to score each site type (i.e. donor, etc.) Several types of features, characterizing statistical and contextual properties within a (-50,+50) nt sequence window flanking each candidate site, are extracted. For example, if we denote the window as S , the basic feature $S_i=G$ would indicate a G in position i , the conjunction $S_i=G \wedge S_{i+1}=T$ would indicate a GT was found in consecutive positions $i, i+1$, and the sparse conjunction $S_i=G \wedge S_j=C$ would indicate bases G,C found in non-consecutive bases i, j . Higher order conjunctions can be included at computational expense, and discretized numerical features are also possible. Details of feature generation, training, and scoring for SNoW-based site classifiers are described in (Chuang and Roth, 2001b).

Initially, the network is connectionless. Connections (weighted edges) between a target node and different features are made in a data-driven manner. When a specific feature is seen in the context of a target node, a connection is built with a default weight. Alternatively, it is possible to make the connection only after a feature is seen a certain number of times. Thus, each target node maintains connections to (potentially) a distinct subset of the feature space.

During training, an example – represented as a list of active features – is presented to the network. The target nodes sum the weights from input nodes corresponding to active features. Let $\mathcal{A}_t = \{i_1, \dots, i_m\}$ be the set of features that are active in an example and are linked to the target node t . Then the linear unit corresponding to t is *active* iff the sum exceeds its threshold θ_t :

$$\sum_{i \in \mathcal{A}_t} w_i^t > \theta_t,$$

where w_i^t is the weight on the edge connecting the i th feature to the target node t .

The Winnow update rule is mistake-driven and has, in addition to the threshold θ_t at the target t , two update parameters: a *promotion* parameter $\alpha > 1$ and a *demotion* parameter $0 < \beta < 1$. These are used to update the current representation of the target t (the set of weights w_i^t), in a multiplicative fashion, only when a mistake in prediction is made.

Winnow exhibits nice theoretical properties for efficiently learning any linear threshold function (Littlestone, 1988). It makes no assumptions of attribute independence and behaves robustly in the presence of redundant or irrelevant features (Littlestone, 1991, 1995; Herbster and Warmuth, 1995). In addition, compared to standard probabilistic and additive update algorithms, multiplicative update algorithms behave much nicer in high dimensional spaces in which the number of relevant features is small relative to the dimensionality. Their generalization properties scale logarithmically with the dimensionality of the feature space and less than linearly with the number of relevant features (Kivinen et al., 1997). As such, it provides the opportunity to learn higher than linear discriminators by increasing the dimensionality of the feature space (for example, by forming conjunctions of basic features) and attempting to learn a linear separator in the augmented feature space.

To score a splice site (i.e. in a test set), the candidate site is presented to a trained network. Each of the target nodes first sums the weights corresponding to features active in the current site, and a sigmoid is applied to the total weight. The result is called the *activation* of the target node, denoted as $\text{act}(t)$. The *confidence* of the site being true is then computed as the ratio:

$$\frac{\text{act}(true)}{\text{act}(true) + \text{act}(pseudo)}.$$

It can be verified that the resulting values are monotonic with the confidence in the prediction Carlson et al. (2001), a crucial fact to the way SNoW scores are used later on by DAGGER. Experimentally, candidates with higher confidence scores generally have a higher probability of being a true site (Chuang and Roth, 2001b).

[†]available at <http://L2R.cs.uiuc.edu/~cogcomp/>

GENE PREDICTION BY DAGGER

Candidate sites and their corresponding SNoW scores from the previous step are passed to DAGGER, which combines the information in a coherent fashion to represent gene structure constraints as a DAG and allows for optimizing a gene structure level cost function. In the following subsections, we first describe how the graph is constructed. This is followed by a description of how edge weights are chosen so that the shortest path represents the gene structure prediction. Some preprocessing steps (filtering, transformation) are also performed on the raw SNoW scores (Chuang and Roth, 2001a).

Graph Construction: Vertices

DAGGER examines the site list to build a weighted DAG. This graph contains one source and one sink vertex corresponding to the beginning and end of the sequence, respectively. Other vertices in the graph are generated from the candidate sites. In general, each start or stop codon corresponds to a single vertex s or t , respectively. In contrast, each candidate donor d is represented by three partial codon vertices $\{d_0, d_1, d_2\}$. Analogously, the vertices $\{a_0, a_1, a_2\}$ are generated for each potential acceptor a .

Graph Construction: Edges

Directed edges in the DAG are added so that every source-sink path represents a valid gene parse conforming to basic gene structure constraints, specifically, no in-frame stops, frame consistency, path length divisible by three, no overlapping edges, and proper ordering of sites: source \rightarrow start \rightarrow (donor \rightarrow acceptor) $_n \rightarrow$ stop \rightarrow sink, $n \geq 0$.

Edges joining two vertices fall in two major classes: coding and non-coding edges. Coding edges represent potential exons, whereas non-coding edges refer to potential introns and regions outside of the gene start-stop boundaries. DAGGER only looks for coding regions of exons, and does not attempt to recognize non-coding exons and 5' or 3' untranslated regions. Thus, "initial exons" are defined to be the subsequence between the true ATG start and its downstream donor site, and "terminal exons" are defined to be the subsequence between the true stop codon and its upstream acceptor. Correspondingly, a single exon gene would be delineated by its start and stop codons. Four types of coding edges are added to the graph:

- Initial exon edges from a start s to a downstream donor d_i are added as long as there is no in-frame stop in between. i is chosen to be the length of the 3' partial codon that would be created if the exon was translated in isolation.
- Similarly, terminal exon edges to a stop t from an upstream acceptor a_i are added as long as there is no in-frame stop in between, and i is chosen accordingly.

- Unspliced ORF edges from a start s to a downstream stop t are added as long as there is no in-frame stop in between them.
- Finally, internal exon edges from an acceptor a_i to a donor d_j are considered by assuming that the exon is translated in each of the three frames $f \in \{0, 1, 2\}$. i and j are chosen to represent the length of the 5' and 3' partial codons that would result from translating the exon in frame f . In general, if $len = (d + 1) - a$ is the exon length, i and j are chosen such that $(i + j) = len \bmod 3$. The edge between a_i and d_j is added if there is no in-frame stop codon in the corresponding frame f .

Subsequently, three types of non-coding edges are added:

- Intron edges between a donor d_i and a downstream acceptor a_j are added if the following criteria are satisfied:
 - Frame constraint: $(i + j) \bmod 3 = 0$. Since i and j are the 3' and 5' partial codon lengths of two adjacent exons, we need to preserve reading frame when the exons are spliced together.
 - The junction formed by splicing together the two partial codons does not form a stop codon. By construction, this junction is always length 0 or 3 and is in-frame.
 - d_i has an in-neighbor and a_j has an out-neighbor. This ensures that the potential acceptor and donor can be part of a valid gene parse. If d_i does not have an in-neighbor, that means the previous steps did not find any upstream site (start or acceptor) that can form an exon bordered by d_i . So d_i will not be a splice site in any predicted gene structure. Similar reasoning applies for a_j . In fact, these vertices can be deleted at this stage.
- Edges between the source and a start vertex represent the non-coding region before a start.
- Edges between a stop and the sink vertex represent the non-coding region after a stop.

Edge Length Cutoffs

One additional (artificial) constraint is that edges are created only if the length of the corresponding subsequence falls within a certain range. One negative consequence is that DAGGER may miss the rare exons and introns of extreme lengths. Furthermore, although each path in the DAG represents a possible gene structure, the length constraints lead to a representation where not all possible gene structures correspond to a source-sink

path. However, the benefit is that the time complexity of graph construction is reduced from quadratic to linear in the number of vertices (i.e. $< 3 \cdot \#sites$). For spliceable exons, the current limits are $min_e = 3$ and $max_e = 600$. For introns, the current limits are $min_i = 35$ and $max_i = 8000$.

Edge Weights and Shortest Paths

Every source-sink path can be interpreted as segmenting the sequence into disjoint coding and non-coding subsequences. At this stage, the goal is to assign edge weights $w(e)$ for each edge e so that the shortest source-sink path corresponds to the “correct” gene structure. Strictly speaking, alternative splicing may generate several correct answers, so a more ideal objective might be to choose $w(e)$ so that correct variants concentrate within the k shortest paths. Since this is currently beyond the scope of this work, we try to choose the edge weight function $w(e)$ to maximize prediction accuracy against annotated exons. $w(e)$ should incorporate the site scores and a coding statistic. The coding statistic is used to distinguish among three possible edges between vertices for a donor d and acceptor a . Although the three edges are incident to distinct partial codon vertices, they would have identical site scores for d and a .

A functional form for w is chosen and parameterized, and then parameters are estimated by optimization techniques. One disadvantage of this approach is that the best functional form is not necessarily obvious and instead may have to be determined empirically. We have experimented with several forms and one that works well has the form:

$$w(e) = \begin{cases} p_1 \log(p_2 \text{conf}(e)) + p_3 \text{hex}(e), & \text{if } e \text{ is a coding edge;} \\ p_4 \log(p_5(1 - \text{conf}(e))) + p_6 \frac{\text{hex}(e)}{\text{length}(e)}, & \text{if } e \text{ is a non-coding edge;} \end{cases}$$

where the edge e has endpoints u and v , $\text{conf}(e) = \text{conf}(u) \cdot \text{conf}(v)$, and the coding statistic $\text{hex}(e)$ is calculated from 5th order inhomogeneous 3-periodic Markov probability tables (Borodovsky and McIninch, 1993). These probabilities are estimated using separate high and low GC content partitions of the training set as described by Burge and Karlin (1997) and transformed into log-odds form. The $\log(\cdot)$ term in w is set to a extreme negative value if its argument is nonpositive.

In this work, parameters \vec{p} were chosen by optimization to a subset of the training set using the Nelder-Mead downhill simplex method (Nelder and Mead, 1965; Wright, 1996). The objective was to maximize AVG , the average of exact exon prediction sensitivity and specificity (defined below). Equivalently, we tried to minimize $-AVG$. Although the Nelder-Mead method is not robust

and is not guaranteed to converge to a minimizer (Wright, 1996), it is easy to implement and does not require derivative information- an advantage since derivatives are not easily calculated in this case. In practice, we were able to achieve noticeable improvement using this method. Further details are described in Chuang and Roth (2001a).

A shortest path tree rooted at the source node was computed using the standard topological sorting / depth first search based algorithm (Cormen et al., 1990). The k shortest paths were subsequently generated using a path deletion algorithm described by Azevedo et al. (1993).

EXPERIMENTAL EVALUATION

Data Sets

BG570 is the benchmark test set prepared by Burset and Guigo (1996) in a large study comparing the performance of several gene finding programs. This set contains 570 vertebrate multi-exon genes; 176 of these are human genes. **KR285** is a non-redundant data set of 285 human multi-exon genes prepared by Kulp and Reese (Kulp et al., 1996; Reese et al., 1997) which has been cleaned of sequences sharing more than 50% identity at the amino acid level using BLASTP (Altschul et al., 1990). (The initial version of the data set appears to have had 304 genes, but had been later reduced to 285 genes after removing suspected pseudogenes). **BK380** is the Genscan learning set of 380 human genes (Burge and Karlin, 1997). 238 of these are multi-exon genes from KR285 and the remaining 142 are single-exon genes.

Note that genes with annotations indicating alternative splicing were removed by the data set creators. In addition, when estimating 5th order Markov probabilities for the coding statistic (see above), we supplemented the training set with the coding regions of 1742 sequences from RefSeq (Pruitt et al., 2000), for a total of $\sim 3150\text{kb}$. This collection is analogous to the 3195kb coding region set used by Burge and Karlin (1997) to estimate the same probabilities. We cleaned RefSeq with respect to itself and against BG570 and KR285 before selecting the final sequences. The RefSeq sequences were not used during other stages of training.

Accuracy Measures

We use standard specificity (sp) and specificity (sn) measures of prediction accuracy (Burset and Guigo, 1996):

- Base-level accuracy (lower case): sn is the fraction of coding bases correctly predicted as coding. sp is the fraction of bases predicted to be coding which are actually coding. The correlation coefficient cc can be used as a summary measure of base-level accuracy. However, since cc is sometimes undefined,

an approximate correlation ac is commonly used instead (Burset and Guigo, 1996).

- Site-level accuracy (mixed case): Sn is the fraction of true sites correctly identified as true sites. Sp is the fraction of predicted true sites which are actually true.
- Exon-level accuracy (upper case): SN is the fraction of true exons predicted correctly. SP is the fraction of predicted exons which are correct predictions. A predicted exon is correct if it corresponds *exactly* to an annotated exon. It is common to use the average $AVG = \frac{SN+SP}{2}$ as a summary measure of exon-level accuracy. Also, ME is the fraction of true exons which have no overlap with any predicted exons, and WE is the fraction of predicted exons which have no overlap with any true exon.

For gene-level accuracy, a gene prediction is said to be correct if it matches the sequence annotation exactly with respect to coding and non-coding boundaries.

When compiling accuracy measures for a test set, it is possible to average over all bases and/or exons in the sequence set (global average), or alternatively, compute the measures for each gene, and then take their averages (per-gene average). We have used per-gene averaging to be consistent with Burset and Guigo (1996), although we typically have slightly better results using a global average.

RESULTS

Evaluation on the BG570 Test Set

To compare DAGGER to other *ab initio* gene finding programs, we evaluated base-level and exon-level accuracy on the BG570 benchmark dataset. SNoW classifiers were trained using the BK380, and edge weights were optimized using a random subset of multi-exon genes. Results for this test set are presented in Table 1. For comparison purposes, results for the programs evaluated by Burset and Guigo (Burset and Guigo, 1996) and also for subsequent programs (MORGAN, VEIL, Genie, HMMgene, Genscan) are reproduced in the table. Of the previous *ab initio* programs, Genscan had the highest accuracy on the BG570 benchmark.

The DAG shortest paths optimization approach in DAGGER compares favorably with other methods. Specifically, DAGGER’s base-level ($ac=0.89$) and exon-level accuracy ($AVG=0.77$) on BG570 exceeded most of the other *ab initio* programs listed in Table 1. DAGGER’s accuracy was slightly lower than Genscan’s, and was competitive with HMMgene (which had higher specificity but lower sensitivity than DAGGER), and other HMM-based gene finders. In addition, although DAGGER does not yet incorporate database homology information, its accuracy

was even comparable to programs that do. Although there are some caveats (see below) with comparing gene finding program accuracy on BG570, the results seemed promising enough to warrant further investigation.

Cross Validation on the KR285 Data Set

One concern about BG570 is that it contains some redundancy; for example, several dozen members of the globin family can be found within. In addition, there is overlap between the BG570 and the Genscan (and also Genie and HMMgene) training set, and there may also be overlap with the training sets of other programs. This was an unavoidable problem when testing on BG570 since different gene finders could not easily be retrained. For these reasons, we wanted to examine DAGGER’s performance on a data set where train and test set overlap could be minimized.

We evaluated DAGGER using seven-fold cross validation on KR285, a non-redundant set of 285 human multi-exon genes. Each of the seven partitions were tested after training on the other six. The combined results are presented in Table 2. For comparison purposes, testing results for several versions of Genie and HMMgene are included. Unfortunately, these are the only published results for this data set that we are aware of. However, since Genie and HMMgene were among the higher accuracy programs on BG570, it was still worthwhile to evaluate DAGGER on this data set.

DAGGER’s base-level ($ac=0.86$) and exon-level ($AVG=0.73$) performance in this cross-validated experiment was competitive with published results for both the *ab initio* and hybrid versions of Genie, and also to several training variants of HMMgene, which had higher specificity but lower sensitivity than DAGGER. We note that HMMgene’s results were reported for a different version of this data set (containing 353 genes) and used 10-fold cross validation. Despite this difference, the two versions of the data set should be similar enough to justify the rough comparison given here. Taken as a whole, the results in Table 2 are consistent with the BG570 test set evaluation, and suggest that DAGGER is competitive with HMM-based approaches.

Providing Partial Information

DAGGER’s predictions on the genes in BG570 and KR285 were also evaluated at the level of site accuracy (Table 3). In BG570, donor Sn and Sp were about 87%, and acceptor Sn and Sp were about 85%. Stop site accuracy was slightly lower, while start sites were the most troublesome (71%). A similar trend was seen in KR285, although general accuracy was lower overall.

Due to gene constraints enforced during graph construction, site prediction mistakes can lead to other prediction mistakes at other sites. For example, if out-of-frame

Program	Base Accuracy				Exon Accuracy					ref.
	sn	sp	ac	cc	SN	SP	AVG	ME	WE	
DAGGER	0.91	0.92	0.89	0.89	0.77	0.77	0.77	0.09	0.09	
Genscan	0.93	0.93	0.91	0.92	0.78	0.81	0.80	0.09	0.05	Burge and Karlin (1997)
HMMgene	0.88	0.94	nr	nr	0.74	0.78	0.76	0.13	0.08	Krogh (1997)
Genie	0.78	0.84	0.77	nr	0.61	0.64	0.62	0.15	0.16	Reese et al. (1997)
VEIL	0.83	0.72	0.73	0.73	0.53	0.49	0.51	0.19	nr	Henderson et al. (1997)
MORGAN	0.81	0.83	0.79	0.79	0.59	0.59	0.59	0.17	nr	Salzberg et al. (1998)
FGENEH	0.77	0.88	0.78	0.80	0.61	0.64	0.64	0.15	0.12	Burset and Guigo (1996)
GeneID	0.63	0.81	0.67	0.65	0.44	0.46	0.45	0.28	0.24	Burset and Guigo (1996)
GeneParser2	0.66	0.79	0.67	0.65	0.35	0.40	0.37	0.29	0.17	Burset and Guigo (1996)
GenLang	0.72	0.79	0.69	0.71	0.51	0.52	0.52	0.21	0.22	Burset and Guigo (1996)
GRAIL 2	0.72	0.87	0.75	0.76	0.36	0.43	0.40	0.25	0.11	Burset and Guigo (1996)
SORFIND	0.71	0.85	0.73	0.72	0.42	0.47	0.45	0.24	0.14	Burset and Guigo (1996)
Xpound	0.61	0.87	0.68	0.69	0.15	0.18	0.17	0.33	0.13	Burset and Guigo (1996)
Genie (Hom)	0.95	0.91	0.91	nr	0.77	0.74	0.76	0.04	0.13	Reese et al. (1997)
GeneID+	0.91	0.91	0.88	0.88	0.73	0.70	0.71	0.07	0.13	Burset and Guigo (1996)
GeneParser3	0.86	0.91	0.86	0.85	0.56	0.58	0.57	0.14	0.09	Burset and Guigo (1996)

Table 1. Base and exon level accuracy on the BG570 benchmark set. ref=source of results, nr=not reported. In bold are summary measures for base-level (*ac*) and exon-level (*AVG*) accuracy, but the other measures can also be used for comparison. The four groups of results correspond to: 1) DAGGER, 2) HMM-based programs, 3) other *ab initio* programs, 4) hybrid programs (GeneID+, GeneParser3, Genie (Hom)) incorporating homology information. MORGAN results are reported for 20% of the data set after training on the other 80%. VEIL results are from a five-fold cross validation experiment. For HMMgene, *AVG* was not reported, so we have estimated it based on *SN* and *SP*. Programs incorporating homology were tested only on the 478 genes less than 8kb in length.

Program	Base Accuracy			Exon Accuracy					ref.
	sn	sp	ac	SN	SP	AVG	ME	WE	
DAGGER	0.87	0.90	0.86	0.71	0.75	0.73	0.14	0.09	
Genie	0.82	0.81	0.79	0.65	0.64	0.64	0.14	0.21	Reese et al. (1997)
Genie (Hom)	0.86	0.85	0.83	0.69	0.68	0.68	0.12	0.18	Reese et al. (1997)
HMMgene (ML est.)	0.81	0.78	nr	0.58	0.65	0.62	0.24	0.15	Krogh (1997)
HMMgene (CML est.)	0.79	0.95	nr	0.61	0.82	0.72	0.27	0.04	Krogh (1997)
HMMgene (CML est. w/ 1-best)	0.82	0.94	nr	0.64	0.79	0.72	0.23	0.06	Krogh (1997)
HMMgene (G/C adaptation)	0.85	0.93	nr	0.69	0.76	0.73	0.17	0.09	Krogh (1997)

Table 2. Base and exon level accuracy after cross-validation on KR285. ref=source of results, nr=not reported. Four training methods for HMMgene are shown (see Krogh (1997) for details). Genie (Hom) is the hybrid version which incorporates database homology information. For HMMgene, *AVG* was not reported, so we have estimated it from *SN* and *SP*.

pseudo-site *X* is wrongly predicted as an exon boundary, it is likely that this mistake will force a mistake at another site *Y* to compensate for the frame shift, or even affect the number of predicted exons.

As described above, site-level accuracy analysis indi-

cated that the start and stop boundaries were the most difficult to predict. So it was of interest to see how much accuracy would improve if DAGGER was provided with partial information- the position(s) of the true start, stop, or both. This information was used to increase

site type	BG570		KR285	
	Sn	Sp	Sn	Sp
starts	0.71	0.71	0.69	0.69
donors	0.87	0.87	0.81	0.87
acceptors	0.86	0.85	0.77	0.84
stops	0.84	0.84	0.78	0.78

Table 3. DAGGER site-level sensitivity (Sn) and specificity (Sp) for the BG570 and KR285 benchmarks.

Given	BG570				KR285			
	ac	AVG	ME	WE	ac	AVG	ME	WE
none	0.90	0.77	0.09	0.09	0.86	0.73	0.14	0.09
+both	0.96	0.92	0.03	0.01	0.93	0.86	0.08	0.02
+starts	0.94	0.87	0.06	0.04	0.90	0.80	0.12	0.05
+stops	0.93	0.83	0.06	0.06	0.90	0.78	0.10	0.05

Table 4. Base and exon level accuracy on BG570 and KR285 when partial information (+starts, +stops, +both) is given (see text for details). The baseline (Given=none) from Tables 1 and 2 are included for reference.

competing edge weights so that paths not passing through the "known" start and/or stop site would have large total weight.

Base- and exon-level prediction results when DAGGER is provided with information on starts and/or stops are presented in Table 4. Providing either starts or stops significantly increases both levels of accuracy, with further improvement when both are known. A liberal interpretation of this form of "cheating" is that it could give an estimate of DAGGER's accuracy if 5' and/or 3' EST information was available and used to restrict the gene boundaries. However, it more likely gives a rough upper bound on DAGGER's accuracy, since EST information is not always available and interpretation is sometimes vague (due to contamination and other factors).

Gene-level Accuracy

We also examined DAGGER's gene-level accuracy by looking for instances where the shortest path outlined the structure of the coding region "perfectly" (i.e. all coding regions predicted exactly). This criteria was the most stringent, so we expected that DAGGER's gene level accuracy would be far from perfect. Since efficient algorithms for ranking suboptimal paths exist (Azevedo et al., 1993; Martins and Santos, 2000), we were able

to address this issue more generally by looking at the k shortest paths for a given gene. We can then ask what fraction of genes have a perfect prediction within one of the k shortest paths. Figure 1 plots this proportion as a function of k for the BG570 and KR285 benchmarks.

For BG570, 256/570 (45%) of the genes were predicted correctly in the shortest path ($k=1$). However, 396/570 (70%) of the genes had a correct prediction within one of the $k=5$ shortest paths. A subset of genes (25%) appears to be more intractable and is not perfectly predicted within the ten shortest paths. KR285 was a more difficult data set, as only 97/285 (34%) of the genes were predicted exactly. When $k=2$, 125/285 (44%) were predicted correctly. Close to 60% of the genes were correctly predicted by $k=10$. Improvements in gene-level accuracy were observed when partial information was provided (Figure 1).

DISCUSSION

In this paper, we have presented an *ab initio* gene finding program called DAGGER which is based on a framework of high dimensional classifiers, constraint satisfaction, and optimization. The gene finding problem is transformed into an optimization problem on graphs, which can take advantage of established algorithms for finding shortest paths, ranking suboptimal paths, and multidimensional optimization.

DAGGER differs from HMM-based approaches in several ways. While HMMs are based on a generative model and thus make several restricting probabilistic assumptions on the sequences to allow the use of efficient estimation algorithms, DAGGER can be viewed as a *conditional model*. It makes no independence or other assumptions on the data. Instead, it is based on directly identifying properties of the input sequence – the classification stage – and then enforcing domain-specific constraints on these. This latter stage is done by optimizing a global cost function which corresponds to these constraints. In principle, probabilistic constraints implicitly utilized within an HMM framework can also be encoded within this framework. Training the site classifiers with the Winnow multiplicative update algorithm in SNoW allows the use of a large number of features in the site prediction and, consequently, provides fairly reliable site identification (Chuang and Roth, 2001b). Within the DAG-based integrated gene model, edge weights are optimized with the pragmatic objective of maximizing exon-level prediction accuracy. Other objectives, which might emphasize a different measure of accuracy, can be used. One disadvantage is that interpreting the information within the trained SNoW classifiers – linear threshold functions – is not as straightforward as interpreting those commonly used within the HMM framework, as SNoW classifiers take into account a large number of features.

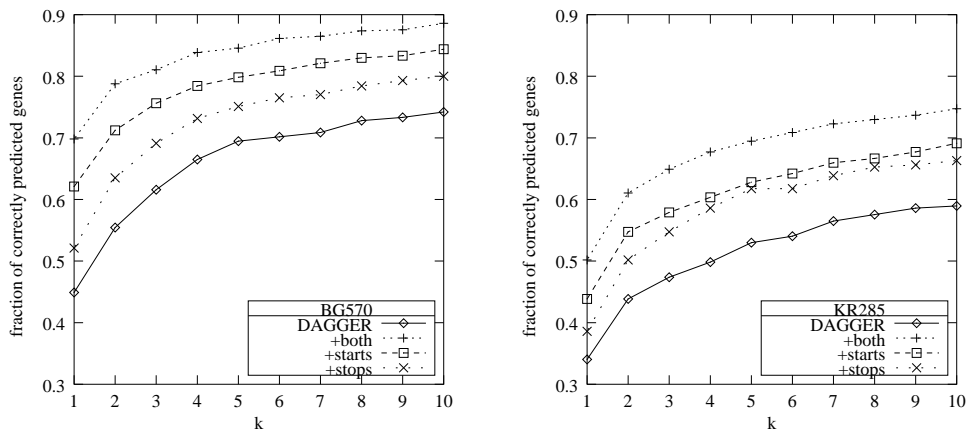


Fig. 1. Fraction of genes in BG570 and KR285 that were predicted correctly within the k th shortest path. The lower curves show DAGGER's baseline accuracy. The other curves (+starts, +stops, +both) show accuracy when partial information is given (see text for details).

Some progress made with sophisticated pruning of the representation Carlson et al. (2001) might simplify this problem.

DAGGER's objective during the optimization stage is to maximize exon prediction accuracy. In this sense it is similar in spirit to HMMgene (Krogh, 1997) which modifies the standard HMM approach. During model estimation, instead of maximizing the probability of the observed sequences given the model, it maximizes the probability of the correct label (of each base as coding, intergenic, or intron). During decoding, HMMgene attempts to approximate the most probable gene prediction, instead of the most probable state sequence.

Evaluation results on two benchmark sets indicate that DAGGER's accuracy is competitive with previous *ab initio* gene finders, including several programs (VEIL, Genie, HMMgene, and Genscan) based on variations of HMMs. Accuracy was similar to that of HMMgene and only a little lower than that of Genscan. The difference between DAGGER's performance and Genscan's may stem from the fact that Genscan incorporates some subtle model details. For example, it includes explicit models of transcription start sites, polyadenylation signals, splice branchpoint regions, as well as exon and intron length distributions and their correlations to GC content. None of these have been modeled in DAGGER. A combination of these factors probably accounts for Genscan's performance edge over DAGGER. Nonetheless, the fact that DAGGER's simple model lags only a little behind Genscan seems encouraging.

Two sources of noise besides obvious annotation errors may limit accuracy. Since most gene finders do not try to predict non-coding exons, there may be true splice sites bordering non-coding exons which are trained as pseudo-

sites. Additionally, a cursory scan of EST-confirmed alternative splicing events from the ISIS database (Croft et al., 2000) suggests that some of the genes in the benchmark sets (which were cleaned of alternative splicing events) are subject to alternative splicing. These alternative splice sites would probably be misannotated in the benchmark data sets.

Future work will concentrate on extending the initial version of DAGGER to accommodate sequences containing partial and/or multiple genes, as this is currently a major limitation. A comparison of the genes for which DAGGER did not predict correctly within the k shortest paths would be useful to determine if these sequences had any common properties which could be exploited. In addition, the partial information experiments suggest that incorporating sequence similarity information from protein, cDNA, and EST sources would be worthwhile. The DAG-based model of gene structure is flexible enough that it should be possible to incorporate other sources of evidence. It will also be of interest to see if objective functions can be designed to predict exons from alternative splicing events, a major source of protein diversity and an increasingly important concern (Black, 2000).

ACKNOWLEDGMENTS

We thank those who constructed the data sets (BG570, BK380, KR285, RefSeq) used in this work. This research is supported by NSF grants IIS-9801638 and IIS-9984168.

REFERENCES

- Altschul, S. F., W. Gish, W. Miller, E. W. Myers, and D. J. Lipman (1990). Basic local alignment search tool. *J. Mol. Biol.* 215, 403–410.

- Azevedo, J. A., M. E. O. S. Costa, J. J. E. R. S. Maderira, and E. Q. V. Martins (1993). An algorithm for the ranking of shortest paths. *European Journal of Operational Research* 69, 97–106.
- Black, D. L. (2000). Protein diversity from alternative splicing: a challenge for bioinformatics and post-genome biology. *Cell* 103, 367–370.
- Borodovsky, M. and J. McIninch (1993). Genmark: parallel gene recognition for both DNA strands. *Comput. Chem.* 17, 123–133.
- Burge, C. and S. Karlin (1997). Prediction of complete gene structures in human genomic DNA. *J. Mol. Biol.* 268, 78–94.
- Burge, C. and S. Karlin (1998). Finding the genes in genomic DNA. *Curr. Opin. Struct. Biol.* 8, 346–354.
- Burset, M. and R. Guigo (1996). Evaluation of gene structure prediction programs. *Genomics* 34, 353–367.
- Carlson, A., C. Cumby, J. Rosen, and D. Roth (1999). The SNoW learning architecture. Technical Report UIUCDCS-R-99-2101, UIUC Computer Science Department.
- Carlson, A. J., J. Rosen, and D. Roth (2001). Scaling up context sensitive text correction. In *Proceedings of the National Conference on Innovative Applications of Artificial Intelligence*.
- Chuang, J. and D. Roth (2001a). Gene recognition based on a DAG shortest path algorithm. Technical Report UIUCDCS-R-2001-2200, UIUC Computer Science Department.
- Chuang, J. and D. Roth (2001b). Splice site prediction using a sparse network of Winnows. Technical Report UIUCDCS-R-2001-2199, UIUC Computer Science Department.
- Cormen, T. H., C. E. Leiserson, and R. L. Rivest (1990). *Introduction to Algorithms*. Cambridge, Mass.: MIT Press.
- Croft, L., S. Schandorff, F. Clark, K. Burrage, P. Arctander, and J. S. Mattick (2000). ISIS, the intron information system, reveals the high frequency of alternative splicing in the human genome. *Nat. Genet.* 24, 340–341.
- Fickett, J. W. (1996). The gene identification problem: An overview for developers. *Comput. Chem.* 20, 103–118.
- Gelfand, M. S., A. A. Mironov, and P. A. Pevzner (1996). Gene recognition via spliced alignment. *Proc. Natl. Acad. Sci. USA* 93, 9061–9066.
- Golding, A. R. and D. Roth (1999). A Winnow based approach to context-sensitive spelling correction. *Machine Learning* 34, 107–130.
- Guigo, R., P. Agarwal, J. F. Abril, M. Burset, and J. W. Fickett (2000). An assessment of gene prediction accuracy in large DNA sequences. *Genome Res.* 10, 1631–1642.
- Haussler, D. (1998). Computational genefinding. *Trends Biochem. Sci., Supplementary Guide to Bioinformatics*, 12–15.
- Henderson, J., S. Salzberg, and K. Fasman (1997). Finding genes in human DNA with a hidden Markov model. *J. Comput. Biol.* 4, 127–141.
- Herbster, M. and M. Warmuth (1995). Tracking the best expert. In *Proc. 12th International Conference on Machine Learning*, pp. 286–294. Morgan Kaufmann.
- Kivinen, J., M. K. Warmuth, and P. Auer (1997). The perception algorithm versus winnow: Linear versus logarithmic mistake bounds when few input variables are relevant. *Artificial Intelligence* 97(1–2), 325–343.
- Krogh, A. (1997). Two methods for improving performance of a HMM and their application for gene finding. In *Proceedings of the 5th International Conference on Intelligent Systems for Molecular Biology*, Menlo Park, pp. 179–186. AAAI Press.
- Kulp, D., D. Haussler, M. G. Reese, and F. H. Eeckman (1996). A generalized hidden Markov model for the recognition of human genes in DNA. In *Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology*, Menlo Park, pp. 134–142. AAAI Press.
- Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning* 2, 285–318.
- Littlestone, N. (1991). Redundant noisy attributes, attribute errors, and linear threshold learning using Winnow. In *Proc. 4th Annu. Workshop on Comput. Learning Theory*, San Mateo, CA, pp. 147–156. Morgan Kaufmann.
- Littlestone, N. (1995). Comparing several linear-threshold learning algorithms on tasks involving superfluous attributes. In *Proc. 12th International Conference on Machine Learning*, pp. 353–361. Morgan Kaufmann.
- Martins, E. Q. V. and J. L. E. Santos (2000). A new shortest paths ranking algorithm. *Investigao Operacional* 20, 47–62.
- Munoz, M., V. Punyakanok, D. Roth, and D. Zimak (1999). A learning approach to shallow parsing. In *EMNLP-VLC'99, Joint SIGDAT Conference on Empirical Methods in NLP and Very Large Corpora*, pp. 168–178.
- Nelder, J. A. and R. Mead (1965). A simplex method for function minimization. *Computer Journal* 8, 308–313.
- Pruitt, K. D., K. S. Katz, H. Sicotte, and D. R. Maglott (2000). Introducing RefSeq and LocusLink: curated human genome resources at the NCBI. *Trends Genet.* 16, 44–47.
- Punyakanok, V. and D. Roth (2001). The use of classifiers in sequential inference. In *NIPS-13; The 2000 Conference on Advances in Neural Information Processing Systems*. MIT Press.
- Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77, 257–286.
- Reese, M. G., F. H. Eeckman, D. Kulp, and D. Haussler (1997). Improved splice site detection in Genie. *J. Comput. Biol.* 4, 311–324.
- Roth, D. (1998). Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and of the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, Menlo Park, pp. 806–813. AAAI Press.
- Roth, D. and D. Zelenko (1998). Part of speech tagging using a network of linear separators. In *COLING-ACL 98, The 17th Int. Conference on Computational Linguistics*, pp. 1136–1142.
- Salzberg, S., A. Delcher, K. Fasman, and J. Henderson (1998). A decision tree system for finding genes in DNA. *J. Comput. Biol.* 5, 667–680.
- Wright, M. H. (1996). Direct search methods: once scorned, now respectable. In *Numerical Analysis 1995 (Proceedings of the 1995 Dundee Biennial Conference in Numerical Analysis)*, pp. 191–208. Addison Wesley Longman.