BIOINFORMATICS

Vol. 14 no. 3 1998
Pages 232–243

GeneGenerator—a flexible algorithm for gene prediction and its application to maize sequences

Jürgen Kleffe[1], Klaus Hermann[1], Wolfgang Vahrson[1], Burghardt Wittig[1] and Volker Brendel[2]

[1]Freie Universität Berlin, Abteilung Molekularbiologie und Bioinformatik, Institut für Molekularbiologie und Biochemie, Arnimallee 22, 14195 Berlin, Germany and [2]Department of Mathematics, Stanford University, Stanford, CA 94305, USA

Received on October 1, 1997; revised on November 24, 1997; accepted on November 27, 1997

Abstract

Motivation: *We developed GeneGenerator because of the need for a tool to predict gene structure without knowing in advance how to score potential exons and introns in order to obtain the best results, pertinent in particular to less well-studied organisms for which suitable training sets are small. GeneGenerator is a very flexible algorithm which for a given genomic sequence generates a number of feasible gene structures satisfying user-defined constraints. The specific implementation described in detail requires minimum scoring for translation start and donor and acceptor splice sites according to previously trained logitlinear models. In addition, potential exons and introns are required to exceed specified minimal lengths and threshold scores for coding or non-coding potential derived as log-likelihood ratios of appropriate Markov sequence models.*

Results: *A database of 46 non-redundant genomic sequences from maize is used for illustration. It is shown that the correct gene structures do not always maximize the considered target function. However, in most cases, the correct or nearly correct structures are found in a small set of high-scoring structures. A critical review of the generated structures sometimes allows the choices to be narrowed by considering additional variables such as predicted splice site strength or local optimality of splice site scores. Summary statistics for prediction accuracy over all 46 maize genes are derived under cross-validation and non-cross-validation training conditions for the Markov sequence models. The algorithm achieved exon sensitivity of 0.81 and specificity of 0.75 on an independent set of 14 novel maize genomic segments.*

Availability: *GeneGenerator runs under Borland-Pascal 7.0 using MS-DOS and C on UNIX work stations. The source code is available upon request.*

Contact: *jkleffe@euler.grumed.fu-berlin-de*

Introduction

The current period of molecular biology and genetics is dominated by large-scale genomic sequencing projects that have already produced the nucleotide-by-nucleotide entire genomes of several bacteria, as well as large contigs of several other model organisms. While there are still many challenging problems with respect to the requisite technology of data acquisition, the focus of research is increasingly shifting to data interpretation. The initial task in this context is annotation of the genomic sequences with respect to their constituent genes and control elements. Several pertinent algorithms have been developed in the last few years and are widely used. The promises and unsolved problems with these approaches have recently been reviewed (Burset and Guigó, 1996; Fickett, 1996; Claverie, 1997). The advantage of these methods is that they give fast and automatic sequence annotations, although a price is paid in terms of reliability. Experimentally determined splicing patterns do not seem to be universally recoverable by optimization of a single scoring function. Furthermore, scoring functions and gene models used in these methods are mostly specific to human or a few other well-studied model organisms and do not apply to other species.

We developed GeneGenerator because of the need for a tool to predict gene structure in maize without knowing in advance how to score potential exons and introns in order to obtain the best results. We therefore focused on maximum flexibility of the basic algorithm. Given a certain genome segment thought to contain a gene, our algorithm produces a set of alternative gene predictions that differ in the assignment of splice junctions. The alternative structures are selected consistent with a number of constraints, the precise specifications of which are at the option of the user. GeneGenerator can list all possible structures satisfying these constraints if the constraints are sufficiently selective or if the

considered sequence is relatively short. Mostly, however, the number of potential gene structures becomes very large due to combinatorial explosion of splice site pairing alternatives. Therefore, many structures that are probably incorrect must be ruled out during the run time of the algorithm. GeneGenerator flexibility includes incorporation of user-defined selection criteria as to which structures to retain. In particular, special implementations of such criteria render the GeneGenerator algorithm equivalent to those of Gelfand and Roytberg (1993), Gelfand *et al.* (1996) or Wu (1996).

While the Gene Generator algorithm can accommodate implementations specific to any particular organisms, here we discuss only applications to maize genes. Splicing in plants is, in many respects, similar to splicing in yeast or animals, but there are also prominent differences (reviewed in Luehrsen *et al.*, 1994; Brown, 1996; Simpson and Filipowicz, 1996). Notably, plant introns are typically of lengths commensurate with the lengths of exons, and very long introns, as occurring in vertebrate genes, appear to be absent. Plant introns are generally U-rich compared to the flanking exons, and this compositional contrast is instrumental in splice site selection and splicing efficiency. The compositional contrast has proved helpful for accurate splice site prediction from sequence inspection (Hebsgaard *et al.*, 1996; Kleffe *et al.*, 1996a). Making appropriate use of these plant-specific properties was the main motivation for the development of GeneGenerator. Together with many useful functions of the DNASTAT package (Kleffe *et al.*, 1995), GeneGenerator forms a toolbox for further development of gene prediction algorithms.

Algorithm

Our gene structure prediction algorithm is similar to those published by Gelfand and Roytberg (1993), Gelfand *et al.* (1996) and Wu (1996), but the details differ. An input sequence is scanned for start codons, stop codons and potential splice sites. At each such site, a current list of partial gene structures is updated according to constraints on the quality of the newly generated exons and introns. For the purpose of this discussion, exons refer to the translated parts of exon sequences only, and genes refer to the part of the pre-mRNA from the translation start codon to the stop codon.

Derivation of gene structures

Genes are represented by strings of exon coordinates in the GenBank CDS line format, called CDS strings. In particular, an initiator codon at sequence positions x to $x + 2$ is represented by '(x,', an exon extending from positions x to y is represented by '$x..y$', introns are represented by ',' and a stop codon at positions $y - 2$ to y is represented by 'y)'. For example, '($x_1..y_1,x_2..y_2,x_3..y_3$)' describes a complete gene with three exons. CDS strings encode gene products that are free of internal in-frame stop codons.

Our algorithm is a simple application of input-driven string expansion rules. It scans the input sequence from the 5′-end to 3′-end. At each stage of the scanning, the currently assembled CDS strings are grouped into seven subsets. Strings ending in '$x.$' are called exon structures and denote potential incomplete genes that end with an exon starting at position x. The exon structures are categorized into three subsets E_p, where p is the reading frame in position x, i.e. $p = 1$, 2 or 3 if x is the first, second or third position of a codon, respectively. Strings ending in '$y,$' are called intron structures, and denote potential genes that end with an exon/intron boundary at position $y/y + 1$. The intron structures are categorized into three subsets I_p, where p is the phase of position $y + 1$ if coding continued into the intron. In other words, p is the phase the next exon has to begin with. Strings ending in 'y)' denote 3′-terminal complete genes ending in an in-frame stop codon at positions $y - 2$ to y. These strings form the set of terminal structures T.

Assume that lists of these strings have been compiled up to a certain position in the sequence. As scanning continues, the lists are updated according to the following expansion rules:

Input	Parent string ending in	Progeny string ending in[a]	Conditions
AG at $x - 2,x - 1$	$y, \in I_p$	$y,x. \in E_p$	
GU at $y + 1,y + 2$	$x. \in E_p$	$x..y, \in I_q$	$q = 1 + (p + y - x) \bmod 3$
AUG at x to $x + 2$		($x. \in E_1$	
Stop at $y - 2$ to y	$x. \in E_p$	$x..y) \in T$	if $(p + y - x) \bmod 3 = 0$

[a]The prefixes of the strings remain unchanged.

Note that the restriction to structures free of internal stop codons allows elimination of the parent string in E_p upon the occurrence of a stop codon and application of the fourth rule above.

We will restrict our discussion to the simplest version of GeneGenerator that is restricted to finding complete single genes represented by start and stop codon-delimited CDS strings. In this case, new CDS strings would only be opened according to the third rule in the above table at AUG locations in the sequence. More generally, the algorithm can easily encompass the presence of 5′- or 3′-terminal incomplete genes for which the start or stop codon, or both, are assumed to occur beyond the bounds of the input sequence. These cases are accommodated by initializing the exon and intron phase p sets E_p and I_p with the strings '.'and ',', respectively, and lead to strings of the form '$.y_1,x_2..y_2,x_3..y_3$)' or '$,x_2..y_2,x_3..y_3,x_4.$', for example, indicating that the sequence is considered to start within an exon in the first case or to start within an intron in the second. The multi-gene version of GeneGenerator employs a different third rule. Each CDS

string in T is extended by ($x.$ and placed back in E_1 upon user-defined conditions on the sequence section which separates the last from the new gene structure. Multi-gene CDS strings look like $(x_1^1..y_1^1, x_2^1..y_2^1, x_3^1..y_3^1)$ $(x_1^2..y_1^2)$.

Constraints on CDS structures

Note that the described algorithm generates all possible intron–exon structures for a given sequence. Because the number of these structures grows exponentially with sequence length, the algorithm incorporates selection criteria that, at each stage in the parsing, limit the string sets to subsets of qualifying structures. These constraints are supplied by means of control functions. The functions DonorQuality and AcceptorQuality determine at each GU or AG dinucleotide whether this site is considered a potential splice site. String expansion takes place if these functions return 'true'. In the simplest case, these functions may be designed to look up lists of pre-estimated splice sites produced by any other program. Another pair of control functions, StartQuality and StopQuality, determine which AUG and stop sites in the sequence are considered possible signals. These four functions comprise first-level control functions. Their decision rules are based on local sequence properties not considering interaction of a site with some particular gene structure. In addition, the functions ExonQuality and IntronQuality discriminate potential exons and introns. These are second-level control functions which determine for each individual CDS string whether rewriting takes place. In these functions, sequence sections can be evaluated based on both local and global sequence properties. Another second-level control function, TerminalQuality, triggers closing of an exon structure by a stop codon and moving it into the set of terminal structures or, alternatively, discarding the structure. Because each structure is completely represented by its CDS string, the decision rules in all second-level control functions may be based upon the complete gene structure assembled up to the current potential sequence junction.

Figure 1 shows the basic flow chart of the complete single-gene version of the GeneGenerator algorithm. Also shown is the logical placement of the only third-level control function, ReduceStructures. This function is invoked if one of the seven sets E_p, I_p or T has reached a predefined maximum size, whereupon it removes the most unfavorable structures to make space in memory for new ones. In our applications, this function is designed to calculate a target function evaluated over each assembled gene structure and to keep only some number of the highest scoring structures. However, the algorithm also allows external selection of favorable structures at this stage in the gene assembly process. For example, comparing the considered sequence with libraries of expressed sequence tags (ESTs) and cDNAs very likely results in a set of exon fragments one may wish to be part of the predicted gene structure. This is done by rejecting all potential splice sites inside such exon fragments and also all introns which overlap these fragments. Preliminary experimental data, such as data from exon trapping or transcript mapping, may be incorporated in similar ways.

Dynamic programming

A special case of the GeneGenerator algorithm obtains if the objective is to maximize additive target functions. The first class of such functions is formed by functions that are additive on the nucleotide level, i.e. sums of scores calculated over all nucleotides of a given sequence. The second class is formed by functions which are additive on the segmental level, i.e. sums of scores calculated over all exons and introns of a given sequence. The latter functions are more general in that they allow the user to combine different segmental measures like splice site strength, length and sequence composition in non-additive ways [see Wu (1996) for a detailed discussion]. These target functions are maximized by a corresponding implementation of the function ReduceStructures. By construction, all partial structures currently held in one of the sets E_p or I_p and sharing the same last splice site may be continued by further introns and exons in the same way. It is, therefore, sufficient to store the best, or the best k structures for each of these subsets. Unfortunately, the number of these subsets grows with sequence length so that the algorithm may require quadratic time unless the control functions ExonQuality and IntronQuality reject exceptionally long introns. GeneGenerator contains a prototype of such a function: ReduceStructures. However, more efficient code is generated if score maximization is implemented into the second-level control functions ExonQuality, IntronQuality and TerminalQuality, too. A k-vector keeping the k best scores for the currently processed partial gene structures can be used to avoid adding low-quality structures to the structure sets which would be removed again by the next call of ReduceStructures.

Additive target functions on the nucleotide level may be maximized in more efficient ways. However, note that the partial structures represented by CDS strings end at different sequence positions upstream of the current position z that GeneGenerator considers at some stage of the prediction process. Therefore, all partial structures of a subset E_p or I_p must be evaluated up to sequence position z before comparison of scores. Additional care is needed for exon structures. The sequence position z is not assigned the same frame for all structures in a considered set E_p because p refers to the frame that the last incomplete exon starts with. Therefore, structures of each set E_p must be divided further into three respective subsets, and the best structures of each set must be retained. Although this appears to be more work, the number of structures to keep no longer grows with sequence length

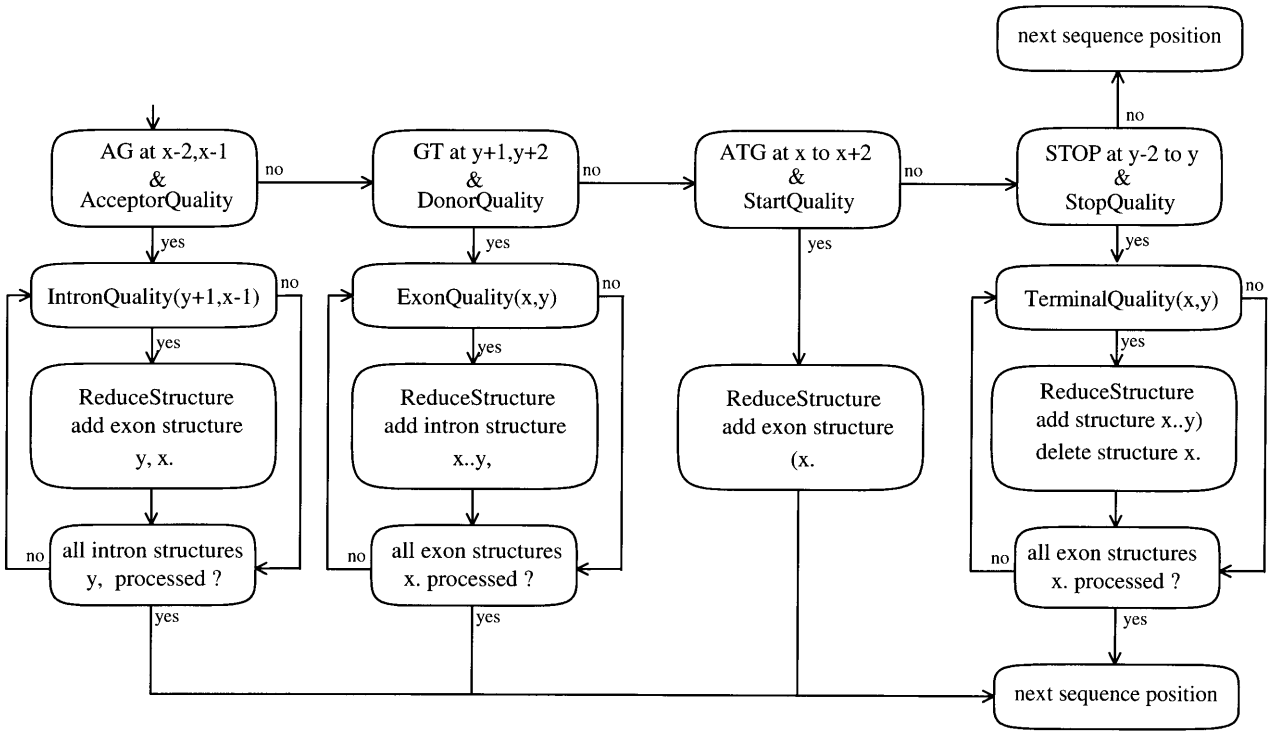


Fig. 1. The GeneGenerator algorithm (simplified).

and the algorithm works in linear time. Constraints on the segmental level, such as minimal length or quality values for exons and introns, are not allowed in this implementation. An example for this type of function ReduceStructures is also available. More generally, GeneGenerator supports the selection of favorable gene structures by allowing one to update a user-defined number of scores from within all second-level control functions. These scores are carried along with the CDS strings. Hence, vector dynamic programming as proposed by Gelfand *et al.* (1996) may also be implemented.

Example of control functions

Here we discuss the implementation of quite simple control functions used to establish a baseline performance of the algorithm.

Level 1. We use the logitlinear models for splice site prediction previously described in Kleffe *et al.* (1996a). The predictions are based on the two variables of (i) degree of matching to the splice site consensus and (ii) local compositional contrast of exon and intron sequence flanks. No consideration is given to predicted splice site strength. Rather, the functions DonorQuality and AcceptorQuality return 'true' for all potential splice sites exceeding the minimal score observed for the real sites in the training set. A logitlinear model was also used to qualify start signal selection by the function StartQuality. Choosing the six nucleotides upstream of the AUG signal as variables, and estimating parameters from 46 maize and 128 *Arabidopsis* sequences (from Kleffe *et al.*, 1996a), the model allows the user to reject ~40% of the false sites without overlooking any true site. The function StopQuality returns 'true' for all termination codons found.

Level 2. The functions ExonQuality, IntronQuality and TerminalQuality were based on Markov models as described previously (e.g. Kleffe *et al.*, 1996b). Let p_k be the probability of a given sequence segment being an exon in frame k (k = 1, 2 or 3), and let p_I be the probability of the segment being an intron. For a potential exon in frame k, ExonQuality and TerminalQuality return 'true' if $p_k > p_I$, and 'false' otherwise. Similarly, IntronQuality returns 'true' if $p_I > p_k$, where k is the appropriate frame of the sequence segment if it were to extend the foregoing exon. An additional requirement was set for a minimum length of four nucleotides for exons and 64 nucleotides for introns. There are considerable problems with fitting frame-dependent Markov models to coding regions of a training sample as small as that for maize. For higher orders, many words of length 6, say, are not observed at all in this set. We therefore estimated models of orders 3, 4 and 5 from word counts supplemented with pseudocounts as described in Lawrence *et al.* (1993). The Appendix gives more detail.

Level 3. The function ReduceStructures, which ranks all CDS strings in a given list, was set to calculate the log-likelihood ratio statistic:

$$LLR = \sum_{j=1}^{m} \log \frac{p(S_j)}{q(S_j)}$$

where m is the number of gene segments S_j (exons and introns), $p(S_j)$ denotes the probability of S_j using the exon and intron models as appropriate, and $q(S_j)$ denotes the probability of S_j based on a mixed Markov model that does not distinguish exons and introns. The latter model was derived from word counts over the entire sequences including exons, introns and intergenic regions. The function ReduceStructures was set up to make GeneGenerator find the 40 best structures by dynamic programming using segmental constraints.

Refinement of structure predictions

Because the output of GeneGenerator consists of a set of alternative structures rather than a single structure prediction, interpretation of the results involves a problem of choice. In the case where GeneGenerator was set up to maximize a target function, one can simply choose the top-scoring structure as the predicted gene. The performance of GeneGenerator will then only depend on the implementation of control functions and is measured by the distance of the top-scoring structure prediction to the correct structure. However, we recommend careful review of at least some alternative structure predictions based on as much additional biological insight as available. Such a scrutiny need not be a simple formal procedure because there is only a small set of structures to be considered. For example, the predicted protein sequences should be queried against protein databases, and predicted exons and partially spliced potential mRNAs may be compared with ESTs and cDNA. With such applications in mind, it is also interesting to study the potential performance of GeneGenerator as measured by the minimum distance to the correct structure taken over all alternative predictions remaining after refinement. The following rather simple formal refinement algorithm for the structures generated by using the control functions described above was used with some success.

(i) The program calculates the number of codons for each structure prediction and cancels all predictions for which this number is less than half the number obtained for the top-scoring structure. This selection mainly removes high-scoring single open reading frames.

(ii) All structure predictions with a target score less than half the maximum score are excluded.

(iii) The remaining structures are categorized by their number of exons, and the resulting subsets are screened individually. Each subset of structures with more than two exons is scanned for local splice site optimality. The structure with the largest number of locally best splice sites is retained, but other structures with the number of locally optimal splice sites less than the number of exons are removed. Then, the strength of splice sites is compared across structures and those with the first donor site, say, scoring less than half of the best first donor site are removed. All splice sites are treated in turn. The remaining structures are sorted for their target scores and form the refined set of predictions which, again, can be evaluated by measuring the distances of the top-scoring or best structure to the correct one.

Implementation

GeneGenerator runs under Borland-Pascal 7.0 using MS-DOS. The source code is available from the first author upon request. GeneGenerator is based on the Pascal unit DNASTAT for the statistical analysis of DNA and protein sequences (Kleffe *et al.*, 1995), and equipped with many functions to form and interpret CDS strings which are useful for developing control functions and refinement algorithms for gene prediction. (A version for UNIX work stations and written in C is now also available).

Gene collections

Genomic sequences from *Zea mays* were retrieved from GenBank and compiled into a specific, non-redundant database as described in Kleffe *et al.* (1996a). Table 1 summarizes the considered set of 46 maize genes, comprising a total of 250 exons and 204 introns. For three donor sites (MZETRNMU intron 2, MZKN1GENE intron 1, ZMU09989 intron 6), GC occurs in place of the consensus GU. These particular GC sites qualify well by formal application of the criterion implemented in the function DonorQuality, but considering each GC site a potential donor leads to an enormous increase in false-positive predictions. There are currently not enough data on such sites to allow specific training of GC donor splice site models. Therefore, we simply changed these three sites to the usual GU consensus in order not to lose the entire sequences for our investigations.

Prediction of gene structures

To set a baseline performance, it is interesting to investigate how well the simple implementation of GeneGenerator is able to recover the 46 considered maize genes. In practical applications, the program is applied to sequences with unknown gene annotation, i.e. to sequences which were not used to form Markov models for exons and introns. This situation is simulated here by removing a given sequence from the training sample and subsequent prediction of its gene structure using Markov models estimated from the reduced set of sequences. These models differ from sequence to se-

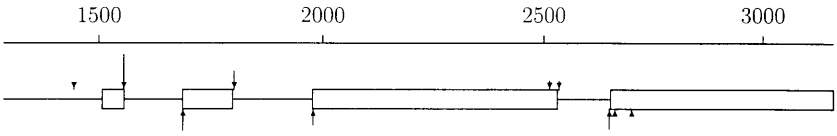
quence and are referred to as cross-validation models. The model derived from all 46 maize and 128 *Arabidopsis* sequences is called a non-cross-validation model. While the cross-validation models more accurately represent the expected performance of the algorithm on truly novel sequences, we consider the results from the non-cross-validation model a reasonable approximation to the expected performance of the algorithm when a much larger training set becomes available. We therefore present results from both approaches and begin with a discussion of three typical examples.

Table 1. Forty-six maize split genes with numbers of introns (Int.)

GeneBank identifier	Accession number	Int.	Function
ZMU14599	U14599	1	bronze-2 gene
ZMALDOAR	X12872	1	anaerobically regulated gene for fructose bisphosphate
ZMBZMCC	X07940 Y00616	1	Bz-McC gene for UDPglucose flavonoid glycosyl-transferase
ZMC2CS	X60205	1	gene for chalcone synthase
ZMLACDEHG	Z11754 S40052	1	gene for lactate dehydrogenase
ZMRAB17G	X15994	1	RAB-17 gene
ZMR28DNA	X59138	1	rab28 gene
ZMGRP	X12564	1	ABA-inducible gene for glycine-rich protein
ZMRBCS	Y00322	1	rbcS gene for ribulose-1,5-bisphosphate carboxylase/oxygenase
ZMB1UB	X52878	1	gene for beta 1 tubulin
MZEOMTH	M73235	1	O-methyltransferase (OMT)
ZMCP1C1G	X81828	1	CYP71C1 gene for cytochrome P-450
ZMZMCIIGE	X87126	1	zmc-II gene
ZMHSFB	X82943	1	mRNA for heat shock factor
ZMU15964	U15964	1	xyloglucan endo-transglycosylase homolog gene
MZETRNMU	M76978	2	transposable element Mu9 DNA, MUDR
MZETRNMU C	M76978	2	transposable element Mu9 DNA, RMUB
MZEMYBAA	M37153	2	locus myb homologue
MZECAT3GN	L05934	2	(CAT3) catalase (Cat3) gene
ZMFNRBP	Z26824	2	gene for Ferredoxin-NADP reductase binding protein
ZMANTG1	X15711	2	gene for mitochondrial adenine nucleotide translocator
ZMALPTUB_1	X15704	3	alpha1-tubulin gene
ZMA1G	X05068	3	A1 gene for 40.1 kDa A1 protein (NADPH-dependent reductase)
MZEACT1G	J01238	3	actin 1 gene (MAc1); intron 3 sequence corrected (K.Luehrsen)
ZMPDCMRNA	X59546	3	Pdc mRNA for pyruvate decarboxylase
ZMU20450	U20450	3	nitrate reductase gene
ZMGPA1	X15408	3	Gpa1 gene for glyceraldehyde-3-phosphate dehydrogenase
ZMCH1GNA	Z22760	3	CHI gene
ZMAUX311	X56737	4	Aux311 gene for auxin-binding protein
ZMGLB1LG	X59083	4	Glb1-L gene for vicilin-like embryo storage protein
ZMAC1	X05424 X05425	4	transposable element Activator (Ac) major transcript
MZKN1GENE	–	4	Knotted-1 gene; hybrid of KN1-2FII and KN1-O genomic seq.
ZMOPA2	X15544	5	opaque-2 gene
ZMTRPA	X76713	5	trpA gene
ZMCATA1	X60135	6	cat1 gene for catalase 1
MZECDPKX	L27484	6	calcium-dependent protein kinase (CDPK) gene
ZMU16123	U16123	6	invertase (Ivr1) gene
ZMADH1FA	X04050	9	alcohol dehydrogenase 1 gene (Adh1-1F)
MZETNENSPM	M25427	9	autonomous transposable element En-1 mosaic protein
ZMPEP	X15642	9	gene for phosphoenolpyruvate carboxylase (EC 4.1.1.31)
ZMGPC1	X15596	10	Gpc1 gene for glyceraldehyde-3-phosphate dehydrogenase
ZMWAXY	X03935 M24258	12	waxy (wx+) locus for UDP-glucose starch glycosyl
ZMSUCS2	X02382	14	sucrose synthase gene
MZEWISHR2	M81603	14	wild-type shrunken-2 allele, exons 1–16
MZECPN60A	L21007	16	nuclear-encoded mitochondrial chaperonin 60 (cpn60I) gene
ZMU09989	U09989	19	D3L H(+)-transporting ATPase (Mha1) gene

Table 2. The top 10 structures predicted for sequence ZMBPA1 along with the log likelihood ratio score, LLR, and the correlation CC with the true structure (calculated as in Burset and Guigó, 1996 Figure 1). Perfect estimation corresponds to CC=1.00. The picture shows the correct structure and all splice site locations occurring in the above structures. Arrows from the top indicate donor sites and arrows from below correspond to acceptor sites. Arrow lengths are proportional to the roots of splice site strength measured by the functions DonorQuality and AcceptorQuality.

Sequence 27 ZMGPA1



LLR	CC	Structure (crossvalidation model)
139.14	1.00	(1509..1553, 1688..1798, 1981..2533, 2653..3155)
137.78	0.95	.....(1739..1798, 1981..2533, 2653..3155)
136.91	0.99	(1521..1553, 1688..1798, 1981..2533, 2653..3155)
132.41	0.98	(1509..1553, 1688..1798, 1981..2514, 2664..3155)
131.06	0.94	.....(1739..1798, 1981..2514, 2664..3155)
130.19	0.98	(1521..1553, 1688..1798, 1981..2514, 2664..3155)
129.41	0.90	(1287..1445, 1688..1798, 1981..2533, 2653..3155)
122.69	0.89	(1287..1445, 1688..1798, 1981..2514, 2664..3155)
117.18	0.96	(1509..1553, 1688..1798, 1981..2514, 2703..3155)
115.82	0.92	.....(1739..1798, 1981..2514, 2703..3155)

LLR	*CC*	Structure (non-crossvalidation model)
183.41	1.00	(1509..1553, 1688..1798, 1981..2533, 2653..3155)
180.96	0.99	(1521..1553, 1688..1798, 1981..2533, 2653..3155)
175.99	0.98	(1509..1553, 1688..1798, 1981..2514, 2664..3155)
175.19	0.95	.....(1739..1798, 1981..2533, 2653..3155)
173.54	0.98	(1521..1553, 1688..1798, 1981..2514, 2664..3155)
170.56	0.90	(1287..1445, 1688..1798, 1981..2533, 2653..3155)
167.78	0.94	.....(1739..1798, 1981..2514, 2664..3155)
163.14	0.89	(1287..1445, 1688..1798, 1981..2514, 2664..3155)
159.37	0.96	(1509..1553, 1688..1798, 1981..2514, 2703..3155)
156.92	0.96	(1521..1553, 1688..1798, 1981..2514, 2703..3155)

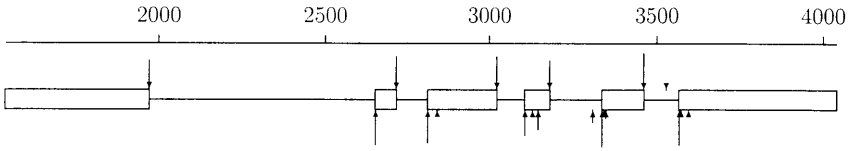
Gpa1 (Table 2). The gene for the A subunit of maize chloroplast glyceraldehyde-3-phosphate dehydrogenase consists of four exons that are contained in the first half of the 6.4 kb GenBank entry ZMGPA1. Table 2 displays the 10 best-scoring gene structure predictions for both cross-validation and non-cross-validation models. In either case, the true structure scores highest. Interestingly, the scores are considerably increased for the non-cross-validation model. Including the considered sequence into the training set increases the probabilities assigned to all of its words. The closer a predicted structure is to the true one, the more words are scored by increased probabilities, leading to the observed higher scores. Structures which have little in common with the true one benefit less from the model change and receive smaller scores.

Opaque-2 gene (Table 3). Table 3 gives the equivalent data for the GenBank entry ZMOPA2. The true gene structure scores best for the non-cross-validation model, but does not score among the 10 best structures for the cross-validation model. In the latter case, the correct intron 5 is rejected by our IntronQuality control function. Reducing second-level control functions to check the minimal length requirements for exons and introns, only, the correct gene structure is found at position 18 in the list of highest scoring gene structures.

Sucrose synthetase gene (Table 4). This example illustrates the problem of combinatorial explosion due to alternative splice site pairings. Cross-validation or not, the highest scoring structures derive from the combinations of two alternatives each for exons 5, 9 and 15, as shown in Table 4. For the non-cross-validation model, the correct structure scores best, whereas for the cross-validation model it gives the eleventh highest score. Given the tight range of the best scores, one can hardly expect a simple

Table 3. The tope 10 structures predicted for sequence ZMOPA2 under cross-validation and non-cross-validation models. See the legend to Table 2 for details.

Sequence 33 ZMOPA2



LLR	CC	Structure (crossvalidation model)
49.63	0.912	(1549..1974, 2653..2713, 2814..3021, 3128..3180, 3343..3528, 3599..3987)
48.83	0.926	(1549..1974, 2653..2713, 2814..3021, 3105..3180, 3339..3528, 3599..3987)
48.67	0.912	(1549..1974, 2653..2713, 2814..3021, 3132..3180, 3339..3528, 3599..3987)
48.44	0.900	(1549..1974, 2653..2713, 2814..3021, 3147..3180, 3339..3528, 3599..3987)
48.41	0.904	(1549..1974, 2653..2713, 2838..3021, 3128..3180, 3343..3528, 3599..3987)
47.61	0.914	(1549..1974, 2653..2713, 2838..3021, 3105..3180, 3339..3528, 3599..3987)
47.44	0.900	(1549..1974, 2653..2713, 2838..3021, 3132..3180, 3339..3528, 3599..3987)
47.22	0.892	(1549..1974, 2653..2713, 2838..3021, 3147..3180, 3339..3528, 3599..3987)
47.17	0.903	(1567..1974, 2653..2713, 2814..3021, 3128..3180, 3343..3528, 3599..3987)
47.09	0.900	(1549..1974, 2653..2713, 2814..3021, 3128..3180, 3310..3528, 3599..3987)

LLR	CC	Structure (non-crossvalidation model)
119.53	1.000	(1549..1974, 2653..2713, 2814..3021, 3105..3180, 3339..3464, 3572..4039)
118.08	0.986	(1549..1974, 2653..2713, 2814..3021, 3132..3180, 3339..3464, 3572..4039)
117.19	0.988	(1549..1974, 2653..2713, 2838..3021, 3105..3180, 3339..3464, 3572..4039)
116.79	0.991	(1567..1974, 2653..2713, 2814..3021, 3105..3180, 3339..3464, 3572..4039)
116.65	0.986	(1549..1974, 2653..2713, 2814..3021, 3128..3180, 3343..3464, 3572..4039)
116.63	0.979	(1549..1974, 2653..2713, 2814..3021, 3147..3180, 3339..3464, 3572..4039)
116.42	0.997	(1549..1974, 2653..2713, 2814..3021, 3105..3180, 3339..3464, 3578..4039)
116.30	0.995	(1549..1974, 2653..2713, 2814..3021, 3105..3180, 3348..3464, 3572..4039)
115.74	0.974	(1549..1974, 2653..2713, 2838..3021, 3132..3180, 3339..3464, 3572..4039)
115.33	0.977	(1567..1974, 2653..2713, 2814..3021, 3132..3180, 3339..3464, 3572..4039)

objective function to be universally successful in assigning the optimal score to the correct gene structure.

Overall performance

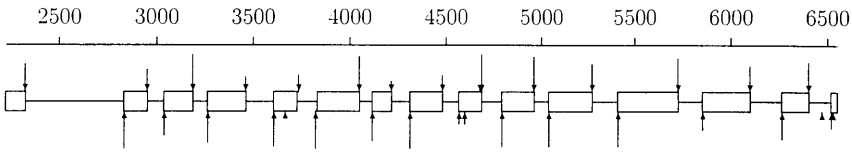
The foregoing examples indicate the sensitivity of gene prediction to parameter estimation and implementation of control functions. It is also evident from the summary statistics given in Table 5. Each line of this table presents averages of performance measures (see Burset and Guigó, 1996; Figure 1) over our 46 maize sequences for different settings of algorithm options. We distinguish order of Markov models (3, 4 or 5), cross-validation or non-cross-validation (Y or N), and two ways of selecting the gene prediction from the 40 or fewer top-scoring alternatives GeneGenerator has provided (T, B). The first choice (T) is the top-scoring prediction. The second choice (B), called best, is one that one does not actually have in gene-finding applications. It tells how well gene prediction would work if we were able to identify from the list of gene predictions the one with the highest correlation coefficient (CC) to the correct structure. In other words, it gives the upper bounds for performance given the set of predictions to choose from. Evidently, the values given in Table 5 show plenty of space for improvement by a second level of scrutiny which would ideally incorporate as much biological insight as available. These potential improvements underline the importance of gene prediction methods which provide sets of alternative and suboptimal solutions.

Refined gene prediction

The described version of GeneGenerator does not distinguish between high- and low-scoring splice sites provided they score above the qualifying threshold. However, it is reasonable to favor among similarly scoring gene predictions those which involve higher scoring splice sites. This approach is aptly illustrated by inspection of the drawings in Table 2–4. For example, the alternative intron 3 for Gpa1 (Table 2) extending from positions 2515 to 2663 gives a similar score for the donor site (0.07 versus 0.06 for the true site at 2534), but a

Table 4. The top 11 (10) structures predicted for sequence ZNSUCS2 under cross-validation and non-cross-validation models. Coordinates are given only for the first exons and for alternatives for exons 5 and 8. See the legend to table 2 for details.

Sequence 43 ZMSUCS2



LLR	CC	Structure (crossvalidation model)
184.58	0.970	(2227..2321, ..., 3675..3733, ..., 4572..4688, ..., 6468..6473)
183.99	0.959	(2227..2321, ..., 3675..3733, ..., 4605..4688, ..., 6468..6473)
183.69	0.989	(2227..2321, ..., 3615..3733, ..., 4572..4688, ..., 6468..6473)
183.35	0.967	(2227..2321, ..., 3675..3733, ..., 4572..4679, ..., 6468..6473)
183.11	0.978	(2227..2321, ..., 3615..3733, ..., 4605..4688, ..., 6468..6473)
182.76	0.956	(2227..2321, ..., 3675..3733, ..., 4605..4679, ..., 6468..6473)
182.51	0.981	(2227..2321, ..., 3675..3733, ..., 4572..4688, ..., 6516..6545)
182.47	0.986	(2227..2321, ..., 3615..3733, ..., 4572..4679, ..., 6468..6473)
181.92	0.971	(2227..2321, ..., 3675..3733, ..., 4605..4688, ..., 6516..6545)
181.88	0.975	(2227..2321, ..., 3615..3733, ..., 4605..4679, ..., 6468..6473)
181.62	1.000	(2227..2321, ..., 3615..3733, ..., 4572..4688, ..., 6516..6545)

LLR	CC	Structure (non-crossvalidation models)
281.20	1.000	(2227..2321, ..., 3615..3733, ..., 4572..4688, ..., 6516..6545)
280.90	0.989	(2227..2321, ..., 3615..3733, ..., 4572..4688, ..., 6468..6473)
279.86	0.981	(2227..2321, ..., 3675..3733, ..., 4572..4688, ..., 6516..6545)
279.81	0.997	(2227..2321, ..., 3615..3733, ..., 4572..4688, ..., 6525..6545)
279.68	0.997	(2227..2321, ..., 3615..3733, ..., 4572..4679, ..., 6516..6545)
279.58	0.990	(2227..2321, ..., 3615..3733, ..., 4605..4688, ..., 6516..6545)
279.56	0.970	(2227..2321, ..., 3675..3733, ..., 4572..4688, ..., 6468..6473)
279.38	0.986	(2227..2321, ..., 3615..3733, ..., 4572..4679, ..., 6468..6473)
279.28	0.978	(2227..2321, ..., 3615..3733, ..., 4605..4688, ..., 6468..6473)
278.47	0.978	(2227..2321, ..., 3675..3733, ..., 4572..4688, ..., 6525..6545)

much worse score for the acceptor site (0.001 versus 0.33 for the true site at 2652). The correct initial exon is bounded by a very strong donor site (score 0.93), whereas the alternative initial exon 1287–1445 is delimited by a very poor donor site (score 0.01). Similar considerations apply to the other examples. In particular, intron 5 of the opaque-2 gene (Table 3), which in the cross-validation study was discarded due to poor intron quality, features very strong splice sites (donor score 0.99, acceptor score 0.94) and is a certain prediction on the basis of splice site quality. For the sucrose synthetase gene (Table 4), the alternative assignments of exons 5 and 15 clearly involve worse splice site selections, and alternatives for exon 9 can be limited to the true assignment of an acceptor site at 4571 (score 0.13) or a downstream site at 4604 (same score). In addition, most structures in Table 4 have a two-codon terminal exon which is rarely seen. Rejecting all structures with lower scoring splice sites brings the correct structure into the leading position. The simple refinement algorithm described in the previous section was developed from such considerations and brings the best solutions on top of the lists for all examples discussed in Table 2–4.

Summary results for the refined sets of gene structure predictions are given in Table 6. They indicate that the proposed way of selecting structures improves average performance. It is, however, unlikely to work well in each individual case. A variety of supplementary features may have to be invoked to give more accurate predictions (e.g. presence of branch point or other motifs). However, because the number of structures to choose from is smaller, it may be easier to reach the performance of the B-option shown in Table 6. After refinement, we have on average five structures to chose from, while we had 38 before refinement.

Table 5. GeneGenerator performance for 46 maize genes listed in Table 1. The parameter settings are coded into the string 'option' as order of the Markov models (3, 4, 5), cross-validation or non-cross-validation conditions (Y, N) and choice of prediction (T, top scoring; B, best). Columns 2–4 provide average sensitivity, specificity and correlation for the prediction of coding nucleotides. Columns 5–8 present average sensitivity and specificity for the prediction of exons and introns as defined in Burset and Guigó (1996), Figure 1

Option	Coding nucleotides			Exons		Introns	
	SN	SP	CC	SN	SP	SN	SP
3_Y_T	0.92	0.88	0.85	0.64	0.61	0.65	0.75
3_Y_B	0.94	0.91	0.89	0.76	0.77	0.74	0.88
3_N_T	0.92	0.89	0.86	0.65	0.61	0.65	0.74
3_N_B	0.94	0.92	0.90	0.77	0.77	0.75	0.88
4_Y_T	0.93	0.92	0.90	0.75	0.73	0.77	0.78
4_Y_B	0.95	0.96	0.94	0.88	0.86	0.88	0.87
4_N_T	0.94	0.98	0.94	0.83	0.77	0.85	0.79
4_N_B	0.96	0.99	0.96	0.95	0.91	0.92	0.91
5_Y_T	0.95	0.92	0.91	0.75	0.72	0.76	0.74
5_Y_B	0.96	0.97	0.95	0.90	0.88	0.93	0.91
5_N_T	0.97	0.99	0.97	0.91	0.86	0.90	0.86
5_N_B	0.98	1.00	0.99	0.97	0.96	0.97	0.97

Table 6. GeneGenerator performance after the selection of preferable structures

Option	Coding nucleotides			Exons		Introns	
	SN	SP	CC	SN	SP	SN	SP
3_Y_T	0.93	0.87	0.86	0.68	0.67	0.71	0.84
3_Y_B	0.94	0.89	0.88	0.74	0.74	0.72	0.87
3_N_T	0.93	0.88	0.87	0.69	0.67	0.70	0.83
3_N_B	0.94	0.90	0.88	0.75	0.75	0.72	0.87
4_Y_T	0.93	0.93	0.90	0.78	0.75	0.80	0.79
4_Y_B	0.95	0.95	0.93	0.83	0.81	0.81	0.84
4_N_T	0.93	0.98	0.94	0.86	0.80	0.89	0.84
4_N_B	0.96	0.98	0.96	0.91	0.88	0.89	0.87
5_Y_T	0.95	0.93	0.92	0.79	0.78	0.80	0.81
5_Y_B	0.96	0.96	0.94	0.87	0.87	0.86	0.89
5_N_T	0.96	0.99	0.97	0.92	0.87	0.90	0.86
5_N_B	0.98	0.99	0.98	0.94	0.94	0.93	0.93

Table 7. Results of GeneGenerator for 14 new maize sequences from GenBank with identifiers MZEPLBH, ZMANTG2, ZMCI31AC3, ZMCYTP450, ZMDNAFER1, ZMGLOSSY, ZMPEPC1G, ZMRUBSMSU, ZMU28017, ZMU29159, ZMU44773, ZMY09238, ZMZAG2 and ZMZMM1

Option	Coding nucleotides			Exons		Introns	
	SN	SP	CC	SN	SP	SN	SP
Unrefined sets of predictions							
3_N_T	0.85	0.93	0.83	0.62	0.52	0.48	0.50
3_N_B	0.95	0.93	0.92	0.78	0.73	0.70	0.76
4_N_T	0.90	0.97	0.90	0.68	0.58	0.60	0.53
4_N_B	0.95	0.97	0.94	0.83	0.78	0.73	0.71
5_N_T	0.92	0.97	0.92	0.76	0.68	0.62	0.59
5_N_B	0.95	0.98	0.95	0.85	0.85	0.76	0.83
Refined sets of predictions							
3_N_T	0.86	0.92	0.84	0.63	0.60	0.68	0.69
3_N_B	0.90	0.92	0.87	0.70	0.69	0.71	0.74
4_N_T	0.91	0.97	0.91	0.80	0.71	0.66	0.63
4_N_B	0.94	0.97	0.93	0.83	0.79	0.74	0.72
5_N_T	0.93	0.97	0.92	0.81	0.75	0.64	0.70
5_N_B	0.95	0.98	0.94	0.84	0.84	0.75	0.81

Performance on an independent test sample

Fourteen new maize sequences with 69 exons were compiled from GenBank, and GeneGenerator was applied to each of them using the model trained on the previous set of 46 sequences. The results are given in Table 7 and show increased error rates. The control functions DonorQuality and AcceptorQuality both reject two true sites, one in each of the sequences with GenBank identifiers ZMCYTP450, ZMDNAFER1, ZMU44773 and ZMY09238. The refined sets of structure predictions using fifth-order Markov models contain the correct solution six times in the top position and two times at the bottom in positions 5 and 7, respectively. In four of the six remaining cases, the closest to correct structures score best among the generated alternatives.

Discussion

Usually, results of gene prediction are presented in terms of sensitivity (SN), specificity (SP) and correlation (CC), as given in Table 5. For the top-scoring prediction under cross-validation conditions, our method reaches a reliable average correlation of 0.85 for third-order Markov models and 0.91 for fifth-order models. However, even with this impressive correlation at the nucleotide level, in many cases the predicted gene structures are not entirely correct. In terms of correctly predicted exon bounds, sensitivity and specificity decrease to 0.75 and 0.72, respectively. In terms of correctly predicted complete genes, only 30 of the 46 maize genes can be found in the lists of the 10 best-scoring structures provided by GeneGenerator, and only 20 are found at the top of these lists. Some improvement was demonstrated by inspection of alternative structures in terms of splice site quality, but further progress must come from the incorporation of supplementary features like comparisons with libraries of ESTs and cDNAs. For this, it is very important to have a choice of structure predictions as provided by GeneGenerator.

By the B-options in Table 5, we have demonstrated how much gene prediction would improve if a proper scrutiny of alternative predictions were able to locate the candidate named best structure that is closest to the correct structure. Such scrutiny may use a complex algorithm incorporating alignments because it evaluates only a small number of alternatives. For our data, it would find 32 genes completely correct and would improve both exon sensitivity and exon specificity, to 0.90 and 0.88, respectively (see line 10 of Table 5). Application of our simple yet imperfect refinement program to the generated structures produced 29 correct solutions in top-scoring position and 30 correct structures among the top 10. It mostly brings more correct structures on top of the lists (see the improved results in Table 6 for the T-option), but also causes loss of the best structure in a few cases so that the B-option results in Table 6 are not as good as in Table 5.

The main advantage of GeneGenerator is the possibility to change control functions easily so that appropriate versions can be applied to each particular problem, possibly in combination with programs for the prediction of regulatory signals and gene location. In the best case of such a combination, we could assume start and stop locations to be known and use GeneGenerator to construct the best structures which extend exactly between these two signals. In such applications, the functions ExonQuality and IntronQuality may be turned off because the number of possible structures is so much smaller. Using fifth-order cross-validation models, the correct structure scored top for 22 sequences and was found among the top 10 in 38 cases. This shows the potential for and the limitation of improving GeneGenerator by more qualified start and stop signal prediction. Also, running GeneGenerator with different sets of control functions on the same sequence may provide new insights. Fickett and Tung (1992) reviewed and compared 13 measures for the coding potential of sequence fragments. Genes with many exons also seem to require the calculation of more than only a few top-scoring structures. Two versions for each of four different splice sites make 16 different structures which may differ little in target value, as seen in Table 4. In such cases, it seems more promising to use a function ReduceStructures which keeps representatives of subsets of similar structures.

The more complex a gene predictor is, the more effort is required to train the algorithm. In effect, gene prediction programs are not available for most species. Good species-independent predictors are not yet known, but the training of our simple version of GeneGenerator is not difficult. Easy-to-use training programs for our control functions are available. We derived successful Markov models for maize exons and introns by modification of *Arabidopsis* models, and suggest that this method of parameter estimation may also be applicable to other small training samples.

Acknowledgements

We thank C.Burge for helpful discussions. V.B. was supported in part by NIH grant 2R01HG00335-09. K.H. was supported by Deutsche Forschungsgemeinschaft Projekt KL 760/1-3 awarded to J.K.

Appendix: Derivation of Markov models

The maize sequence sample described in Table 1 is sufficiently large for estimation of second-order Markov models, and these estimates were used to derive pseudocounts. Precisely, omitting the index for frame to simplify notation, the second-order transition and initial probabilities were derived as:

$$p(B|W_2) = \frac{n(W_2, B)}{n(W_2)}, \quad p(W_2) = \frac{n(W_2)}{N} \qquad (1)$$

where W_2 is the dinucleotide preceding base B, $n(W_2, B)$ is the count of triplets (W_2,B), $n(W_2)$ is the count of dinucleotide W_2, and N is the sum of $n(W_2)$ over all dinucleotides W_2. Note that the second-order transition probabilities are stop codon sensitive, a property which is carried over to all higher order Markov models.

Parameter estimates for higher order Markov models were derived as described in the following for the case of fifth-order models. Estimates based on a count of n pentamers were supplemented with $\sqrt{n}$ pseudocounts. Thus, the probability of base B given it is preceded by pentamer W_5 and dimer W_2 was set to:

$$p(B|W_5) = \frac{n(W_5, B)}{n + \sqrt{n}} + \frac{p(B|W_2)}{1 + \sqrt{n}}, \quad n = \sum_B n(W_5, B) \quad (2)$$

and the initial probability for pentamer W_5 ending in dimer W_2 was set to:

$$p(W_5) = \frac{n(W_5)}{N + \sqrt{N}} + \frac{p(W_2)}{1 + \sqrt{N}}, \quad N = \sum_{W_5} n(W_5) \quad (3)$$

In spite of the pseudocount approach, these models still change notably upon excluding one of the 46 maize sequences from the training set because each sequence in this limited set contributes significantly to the small counts $n(W_5, B)$ of hexamers and $n(W_5)$ of pentamers. We therefore considered stabilized Markov models by utilizing word counts from 128 *Arabidopsis* sequences. In this case, the probability of base B given it is preceded by pentamer W_5 and dimer W_2 is:

$$p(B|W_5) = \frac{n_A(W_5, B) + n_M(W_5, B)}{n + \sqrt{n}} + \frac{p(B|W_2)}{1 + \sqrt{n}} \quad (4)$$

where $\sum_B n_A(W_5, B) + \sum_B n_M(W_5, B)$, and the initial probability for pentamer W_5 ending in dimer W_2 is:

$$p(W_5) = \frac{n_A(W_5) + n_M(W_5)}{N + \sqrt{N}} + \frac{p(W_2)}{1 + \sqrt{N}} \quad (5)$$

where $N = \sum W_5 n_A(W_5) + \sum W_5 n_M(W_5)$. The suffixes A and M stand for *Arabidopsis* and maize, respectively. The effect of each maize sequence on the Markov model is now reduced, but, on the other hand, the model is biased due to the incorporation of *Arabidopsis* sequences which differ from maize in G + C content.

References

Brown,J.W.S. (1996) *Arabidopsis* intron mutations and pre-mRNA splicing. *Plant J.*, **110**, 771–780.

Burset,M. and Guigó,R. (1996) Evaluation of gene structure prediction programs. *Genomics*, **34**, 353–367.

Claverie,J.-M. (1997) Computational methods for the identification of genes in vertebrate genomic sequences. *Hum. Mol. Genet.*, *6*, 1735–1744.

Fickett,J.W. (1996) Finding genes by computer: the state of the art. *Trends Genet.*, **12**, 316–320.

Fickett,J.W. and Tung,C.S. (1992) Assessment of protein coding measures. *Nucleic Acids Res.*, **20**, 6441–6450.

Gelfand,M.S. and Roytberg,M.A. (1993) Prediction of the exon–intron structure by a dynamic programming approach. *BioSystems*, **30**, 173–182.

Gelfand,M.S., Podolsky,T.A., Astakhova,T.V. and Roytberg,M.A. (1996) Recognition of genes in human DNA sequences. *J. Comput. Biol.*, **3**, 223–234.

Hebsgaard,S.M., Kerning,P.C., Tolstrup,N., Engelbrecht,J., Rouzé,P. and Brunak,S. (1996) Splice site prediction in *Arabidopsis thaliana* pre-mRNA by combining local and global sequence information. *Nucleic Acids Res.*, **24**, 3439–3452.

Kleffe,J., Hermann,K., Gunia,W., Vahrson,W. and Wittig,B. (1995) DNASTAT: a Pascal unit for the statistical analysis of DNA and protein sequences. *Comput. Applic. Biosci.*, **11**, 449–455.

Kleffe,J., Hermann,K., Vahrson,W., Wittig,B. and Brendel,V. (1996a) Logitlinear models for the prediction of splice sites in plant pre-mRNA sequences. *Nucleic Acids Res.*, **247**, 4709–4718.

Kleffe,J., Hermann,K. and Borodovsky,M. (1996b) Statistical analysis of GeneMark performance by cross validation. *Comput. Chem.*, **20**, 123–133.

Lawrence,C.E., Altschul,S.F., Boguski,M.S., Liu,J.S., Neuwald,A.F. and Wootton,J.C. (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, **262**, 208–214.

Luehrsen,K.R., Taha,S. and Walbot,V. (1994) Nuclear pre-mRNA processing in higher plants. *Prog. Nucleic Acids Res. Mol. Biol.*, **47**, 149–193.

Simpson,G.G. and Filipowicz,W. (1996) Splicing of precursors to mRNA in higher plants: mechanisms, regulation and sub-nuclear organisation of the spliceosomal machinery. *Plant Mol. Biol.*, **32**, 1–41.

Wu,T.D. (1996) A segment-based dynamic programming algorithm for predicting gene structure. *J. Comput. Biol.*, **3**, 375–394.