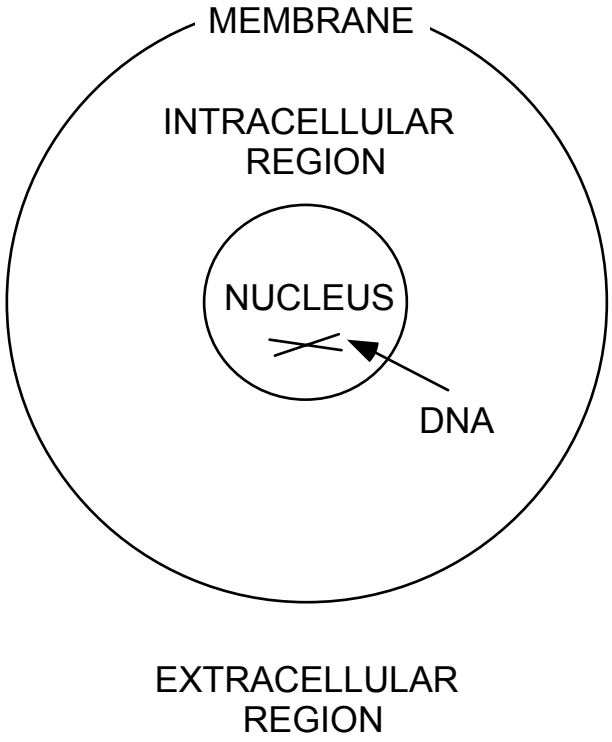
CLASSIFICATION BY CELLULAR LOCATION

FIVE-WAY CLASSIFICATION (E, I, N, M, A)



FIVE-WAY CLASSIFICATION (E, I, N, M, A)

EXTRACELLULAR (E) PROTEIN

172-residue Hemagglutinin/amebocyte aggregation factor precursor (called HAAF_LIMPO in the SWISS-PRO database of proteins)

MNSPAIVIIIFSTLTFSEAWVNDWDGALNFQCQLKDSIKTISS
IHSNHHEDRRWNFGCERTLRDPSCYFTNYVNDWDKLLHFTCKS
GEAIAGFNSYHDNRREDRRWKIYCCKDKNKCTDYRTCAWTGYV
NSWDGDLHYTVPKDYVLTGVISEHDNHREDRRWKFQHCRLKNC

INTRACELLULAR (I) PROTEIN

165-residue Peptidyl-prolyl cis-trans isomerase (CYPH_STRCH)

MTTKVYFDITIDDAPAGRITFNLFDDVVPKTAENFRALATGEK
GFGYAGSSFHRVITDFMLQGGDFTRGDGTGGKSIYGEKFADEN
FQLKHDRVGLLSMANAGKNTNGSQFFITTVLTPWLDGKHVVFG
EVADDDSMALVRKIEALGSSSGRTSAKVTIAESGAL

FIVE-WAY CLASSIFICATION (E, I, N, M, A) – CONTINUED

NUCLEAR (N) PROTEIN

188-residue Prostaglandin-H2-D-isomerase precursor (PGHD_RAT)

MAALPMLWTGLVLLGLLGFPQTPAQGHDTVQPNFQQDKFLGRW
YSAGLASNSSWFREKKELLFMCQTVVAPSTEGGLNLTSTFLRK
NQCETKVMVLQPAGVPGQYTYNSPHWQLPLPLSVETDYDEYAF
LFSKRTKGPGQDFRMATLYSRAQLLKEELKEKFITFSKDQGLT
EEDIVFLPQPDKCIQE

MEMBRANE (M) PROTEIN

163-residue Halocyanin precursor (HCY_NATPH)

MKDISRRRFVLGTGATVAAATLAGCNGNGNGNGNGNGEPDT
PEGRADQFLTDNDALMYDGDITDETGQDEVVVVTGAGNNGFAF
DPAAIRVDVGTTVTWEWTGDGGAHNVVSEPESDFEFESDRVDE
EGFTFEQTFDDEGVALYVCTPHRAQGMYGAVIVE

ANCHORED (A) MEMBRANE PROTEIN

171-residue T-cell surface glycoprotein CD3 delta chain precursor (CD3D_HUMAN)

MEHSTFLSGLVLATLLSQVSPFKIPIEELEDRVFVNCNTSITW
VEGTVGTLLSDITRLDLGKRILDPRGIYRCNGTDIYKDKESTV
QVHYRMCQSCVELDPATVAGIIVTDVIATLLLALGVFCFAGHE
TGRLSGAADTQALLRNDQVYQPLRDRDDAQYSHLGGNWARNK

TYPES OF COMPUTATION

- **Single-residue statistics**
  - **76% accuracy in 5-way classification of cellular location (Cedano, Aloy, Perez-Pons, and Querol in 1997 *Journal of Molecular Biology*).**
  - **77% accuracy in 3-way classificaiton of cellular location based on TERTIARY STRUCTURE (O'Donoghue, and Rost in 1998 *Journal of Molecular Biology*.**
- **Adjacent-pair statistics. Two-residue percentages can improve accuracy (Nakashima and Nishikawa 1994).**
- **Neural networks. Fixed window size and fixed calculation (sum of weighted inputs).**
- **Decision trees - Multi-branch tests of single user-defined attributes.**
- **Inductive logic programming – Horn clauses.**
- **Motifs –A matching program based on regular expressions.**

  `[LIVM]-[LIVM]-D-E-A-D-X-[LIVM]-[LIVM]`

PROGRAMMATIC MOTIFS

• Programmatic motifs are much more than conventional motifs.

• Programmatic motifs bring to bear the large and varied arsenal of computational capabilities inherent in ordinary computer programs, including

- **arithmetic functions**
- **conditional operations**
- **iterations**
- **named memory**
- **indexed memory**
- **set-creating operations**
- **subroutines**

ARCHITECTURE

Each program has a uniform seven-branch architecture, including

- **three set-creating automatically defined functions (`ADF0`, `ADF1`, and `ADF2`),**
- **two arithmetic-performing automatically defined functions (`ADF3` and `ADF4`),**
- **one iteration-performing branch (`IPB`), and**
- **one result-producing branch (`RPB`).**

FUNCTION AND TERMINALS FOR THE THREE SET-CREATING AUTOMATICALLY DEFINED FUNCTIONS (ADF0, ADF1, AND ADF2)

- **The terminal set for the three set-creating automatically defined functions**

$T_{adf\text{-}sc}$ = {(A?), (C?), _, (Y?)}

- **Terminals such as (A?) are the zero-argument residue-detecting function returning a numerical +1 if the current residue is alanine (A) but otherwise returning a numerical –1.**

- **The function set for the three set-creating automatically defined functions**

$F_{adf\text{-}sc}$= {ORN}

FUNCTION AND TERMINALS FOR THE TWO ARITHMETIC-PERFORMING AUTOMATICALLY DEFINED FUNCTIONS (ADF3 AND ADF4)

- **The terminal set for the arithmetic-performing automatically defined functions**

$T_{adf\text{-}ap} = \{(A?), (C?), \_, (Y?), \Re_{bigger\text{-}reals}\}$.

- **The function set for the two arithmetic-performing automatically defined functions**

$F_{adf\text{-}ap} = \{+, -, *, \%, IFGTZ, ORN\}$.

FUNCTION AND TERMINALS FOR THE ITERATION-PERFORMING BRANCH

• The iteration-performing branch provides a mechanism for performing an iterative calculation over the entire protein sequence. The proteins are of different length. The work-performing steps of the iteration-performing branch are performed iteratively for each of the amino acid residues of the given protein.

• The iteration over the protein sequence is indexed by the iterative variable, `INDEX`. At the beginning of each step of the iteration, the `ACTIVE` residue of the protein is the same as iterative variable, `INDEX`. The distinction between the `ACTIVE` residue and the iterative variable, `INDEX`, is important because the `ACTIVE` residue can be incremented inside a given step (`INDEX`) of the iteration using the `FORWARD` function (described below).

FUNCTION AND TERMINALS FOR THE ITERATION-PERFORMING BRANCH – CONTINUED

• Memory cells can be used to store the intermediate and final results during the iterative calculation. There are two types of memory:

- five cells of named memory and
- 20 cells of indexed memory.

• The contents of all cells of named and indexed memory are initialized to zero before execution of the iteration-performing branch for a particular fitness case.

• The terminal set for the iteration-performing branch

T_{ipb} ={ADF0, ADF1, ADF2, ADF3, ADF4,
(A?), (C?), _, (Y?), M0, M1, M2, M3, M4,
READ-RESIDUE, $\Re_{\text{bigger-reals}}$}.

FUNCTION AND TERMINALS FOR THE ITERATION-PERFORMING BRANCH – CONTINUED

• The terminal `M0` returns the value contained in cell `M0` of named memory (and similarly for the four other terminals `M1`, `M2`, `M3`, and `M4`).

• The terminal `READ-RESIDUE` returns the value contained in the cell of indexed memory implicitly indexed by the active residue.

• The function set for the iteration-performing branch

F_{ipb} = {`ORN`, `SETM0`, `SETM1`, `SETM2`, `SETM3`, `SETM4`, `IFGTZ`, `WRITE-RESIDUE`, `FORWARD`, `+`, `-`, `*`, `%`}.

• The one-argument `WRITE-RESIDUE` function sets into the cell of indexed memory implicitly indexed by the active residue to the value of its argument.

FUNCTION AND TERMINALS FOR THE ITERATION-PERFORMING BRANCH – CONTINUED

• The one-argument `FORWARD` function increments the `ACTIVE` residue by one and returns the value of its one argument using the current value of `ACTIVE`. Upon completion of the evaluation of the argument of the `FORWARD` function, the `ACTIVE` residue is restored to its previous value. If the `FORWARD` function attempts to increment the `ACTIVE` residue beyond the length of the protein, the `ACTIVE` residue remains unchanged. Note that the values returned by the five automatically defined functions depend on the `ACTIVE` residue.

FUNCTION AND TERMINAL SET FOR THE RESULT-PRODUCING BRANCH

•The terminal set, T_{rpb}, for the result-producing branch is,

$T_{rpb} = \{$`M0`, `M1`, `M2`, `M3`, `M4`, $\Re_{\text{bigger-reals}}\}$.

• The function set for the result-producing branch

$F_{rpb} = \{$`ORN`, `IFGTZ`, `+`, `-`, `*`, `%`, `READ`$\}$.

• The one-argument `READ` function returns the value stored in the indexed memory cell `J` where `J` is the value of its one argument (adjusted by flooring it and then taking it modulo the size, 20, of the indexed memory).

• The wrapper (output interface) for the result-producing branch consists of the `IFGTZ` function, so a positive return value is interpreted as `YES` and a zero or negative return value is interpreted as `NO`.

FITNESS

• Fitness is measured over a number of protein sequences (fitness cases). Cedano, Aloy, Perez-Pons, and Querol 1997 used 1,000 proteins (200 from each of five classes) as in-sample fitness cases. A substantial amount of computer time (about three-fourths) was saved by randomly choosing 50% of these proteins of length less than 600 as the in-sample fitness cases. This approach yielded 471 in-sample fitness cases (88 membrane, 91 nuclear, 108 intracellular, 124 extracellular, and 70 anchored proteins) having a total of 122,489 residues.

FITNESS — CONTINUED

$$N_{fc} = N_{tp} + N_{tn} + N_{fp} + N_{fn}$$

CORRELATION

$$C = \frac{\sum_j (S_j - \bar{S})(P_j - \bar{P})}{\sqrt{\sum_j (S_j - \bar{S})^2 \sum_j (P_j - \bar{P})^2}}$$

$$C = \frac{N_{tp} N_{tn} - N_{fn} N_{fp}}{\sqrt{(N_{tn} + N_{fn})(N_{tn} + N_{fp})(N_{tp} + N_{fn})(N_{tp} + N_{fp})}}$$

STANDARDIZED FITNESS

$$\frac{1 - C}{2}.$$

CROSS-VALIDATION

• The results were first cross-validated using the remaining proteins of length less than 600.

• The results were further cross-validated using the same 200 out-of-sample fitness cases as in Cedano et al. (1997) (specifically, 41 membrane, 46 nuclear, 55 intracellular, 33 extracellular, and 25 anchored proteins) having a total of 100,029 residues.

TABLEAU FOR CELLULAR LOCATION

Objective:	**Discover a program to classify a protein sequence as to whether or not it resides in a particular cellular location.**
Program architecture:	**One result-producing branch (`RPB`), three set-creating automatically defined functions (`ADF0`, `ADF1`, and `ADF2`), two arithmetic-performing automatically defined functions (`ADF3` and `ADF4`), and one iteration-performing branch (`IPB`).**
Function set for the result-producing branches:	**F_{rpb} = {`ORN`, `IFGTZ`, `+`, `-`, `*`, `%`, `READ`}.**
Terminal set for the result-producing branches:	**T_{rpb} = {`M0`, `M1`, `M2`, `M3`, `M4`, $\Re_{bigger\text{-}reals}$}.**

Function set for the set-creating automatically defined functions:	**$F_{adf\text{-}sc}$ = {ORN}.**
Terminal set for the set-creating automatically defined functions:	**$T_{adf\text{-}sc}$ = {(A?), (C?), _, (Y?)}.**
Function set for the arithmetic-performing automatically defined functions:	**$F_{adf\text{-}ap}$ = {+, –, *, %, IFGTZ, ORN}.**

Terminal set for the arithmetic-performing automatically defined functions:	$T_{adf\text{-}ap} = \{$`(A?)`, `(C?)`, `_`, `(Y?)`, $\Re_{bigger\text{-}reals}\}$.
Function set for the iteration-performing branch:	$F_{ipb} = \{$`ORN`, `SETM0`, `SETM1`, `SETM2`, `SETM3`, `SETM4`, `IFGTZ`, `WRITE-RESIDUE`, `FORWARD`, `+`, `-`, `*`, `%`$\}$.
Terminal set for the iteration-performing branch:	$T_{ipb} = \{$`ADF0`, `ADF1`, `ADF2`, `ADF3`, `ADF4`, `(A?)`, `(C?)`, `_`, `(Y?)`, `M0`, `M1`, `M2`, `M3`, `M4`, `READ-RESIDUE`, $\Re_{bigger\text{-}reals}\}$.

Fitness cases:	**The wrapper (output interface) for the result-producing branch consists of the `IFGTZ` function, so a positive return value is interpreted as `YES` and a zero or negative return value is interpreted as `NO`.**
Raw fitness:	**Correlation *C* (ranging from –1.0 to +1.0).**
Standardized fitness:	**Standardized fitness is 1–*C*/2 (ranging from 0.0 to 1.0).**
Hits:	**Not used for this problem.**
Wrapper:	**The wrapper (output interface) for the result-producing branch consists of the `IFGTZ` function, so a positive return value is interpreted as `YES` and a zero or negative return value is interpreted as `NO`.**

Parameters:	$M = 320{,}000$. $G = 201$. $Q = 5{,}000$. $D = 64$. $B = 2\%$. $S_{rpb} = 400$. $S_{adf} = 40$. $N_{rpb} = 1$. $N_{max\text{-}adf} = 5$. $N_{ipb\text{-}max} = 1$.
Results Designation:	**Best-so-far pace-setting individual.**
Success predicate:	**Fitness appears to have reached a plateau.**

RESULTS FOR NUCLEAR PROTEINS

• On generation 7, genetic programming evolved a two-way classification program for nuclear proteins with 86% in-sample accuracy, 91% out-of-sample accuracy (on-line), and 84% accuracy on the 200 out-of-sample proteins of Cedano et al. (1997).

• The genetically evolved program actually invokes its result-producing branch and iteration-performing branch, but no automatically defined functions. It uses three cells of named memory (`M1`, `M2`, and `M4`) for communication between its iteration-performing branch and the result-producing branch.

RESULTS FOR NUCLEAR PROTEINS – CONTINUED

Performance of best-of-run classifying program for nuclear proteins

	I	E	A	M	N
N	7	4	0	0	26
Other	48	29	25	32	20

- **Most of the false positives are incorrect intracellular classifications. Intracellular proteins, like nuclear proteins, tend to be electrically charged.**
- **No errors were made for either the membrane or anchored membrane proteins.**

RESULTS FOR NUCLEAR PROTEINS – CONTINUED

- **The 36-point iteration-performing branch**

```
(setm2 (setm0 (setm3 (+ (– (p?) (l?))
(setm2 (setm0 (setm3 (+ (setm1 (forward
(– (r?) (v?)))) (setm3 (setm1 (setm3 (+
(– (q?) (v?)) (setm1 (setm0 (setm3 (+
(setm1 (forward (– (r?) (v?)))) (setm3
(setm1 (setm1 m0))))))))))))))))
```

- **The six-point result-producing branch**

```
(+ (ifgtz m4 m1 -3.337654) m2)
```

RESULTS FOR NUCLEAR PROTEINS – CONTINUED

When analyzed, the evolved solution for classifying a protein as a nuclear is:

- Increment the running sum, `M0`, by adding the number in table below to `M0` for each residue of the protein.
- If –3.34 + `M0` > 0, then classify the protein as a nuclear protein; otherwise, classify the protein as non-nuclear.

Increments used for nuclear proteins.

Residue	Increment
R	+4
Q	+2
P	+2
L	–2
V	–6
Other	0

- **DNA carries carries a net negative charge.**
- **The arginine R residue is positively charged while the glutamine residue Q is polar. Proline P, leucine L, and valine V are not electrically charged.**
- **Valine V is hydrophobic**

RESULTS FOR MEMBRANE PROTEINS

• On generation 10, genetic programming evolved a two-way classification program for membrane proteins with 85% in-sample accuracy, 87% out-of-sample accuracy (on-line), and 89% accuracy on the 200 out-of-sample proteins (off-line) of Cedano et al. 1997.

• The iteration-performing branch wrote to indexed memory extensively (in seven different places) and also employed named memory to store intermediate results. One set-creating automatically defined function (`ADF0`) was used. One cell of named memory (`M1`) was then used for communication between the iteration-performing branch and the result-producing branch.

RESULTS FOR MEMBRANE PROTEINS – CONTINUED

Performance of best-of-run classifying program for nuclear proteins.

	I	E	A	M	N
M	2	2	1	24	0
Other	53	31	24	17	46

- **Since a portion of each membrane protein is extracellular and intracellular, membrane proteins are most easily mistaken for extracellular and intracellular proteins. Similarly, membrane proteins may be easily mistaken for anchored membrane proteins.**

- **Membrane proteins are most dissimilar to nuclear proteins and are, accordingly, not misclassified as nuclear at all by this classifying program.**

RESULTS FOR MEMBRANE PROTEINS – CONTINUED

When analyzed, the evolved solution for classifying a protein as a membrane is:

- Increment the running sum, `M4`, by adding the number in below table to `M4` for each residue of the protein.
- For each residue, write the current value of `M4` into the cell of indexed memory whose index equals the current residue.
- If the 0th-cell of indexed is positive, then classify the protein as a membrane protein; otherwise, non-membrane.

Residue	Increment
F	+2
I	+2
P	–2
N	–2
K	–2
Other	0

- **F and I are hydophobic**
- **K is positively charged and N is polar**
- **Possible explanation for P. It's is helix breaker (and many are α-helices)**

RESULTS FOR EXTRACELLULAR PROTEINS

- **On generation 26, genetic programming evolved a two-way classification program for extracellular proteins with 82% in-sample accuracy, 87% out-of-sample accuracy (on-line), and 83% accuracy on the 200 out-of-sample proteins (off-line) of Cedano et al. 1997.**

- **The evolved program referred once to its arithmetic-performing automatically defined function `ADF4` and repeatedly referred to its set-creating automatically defined function `ADF1`. It twice used the `FORWARD` function to examine downstream residues. It uses one cell of named memory (`M0`) for communication between its iteration-performing branch and the result-producing branch.**

RESULTS FOR ANCHORED PROTEINS

• On generation 11, genetic programming evolved a two-way classification program for anchored proteins with 80% in-sample accuracy, 85% out-of-sample accuracy (on-line), and 83% accuracy on the 200 out-of-sample proteins of Cedano et al. (1997).

• The iteration-performing branch uses both indexed memory and named memory. The program actually invokes one set-creating automatically defined function (ADF2) and one arithmetic-performing automatically defined function (ADF3).

CONCLUSIONS – CELLULAR LOCATION

Genetic programming evolved two-way classifying programs for identifying extracellular proteins and membrane proteins using programmatic motifs with

- **84% accuracy for nuclear proteins,**
- **89% accuracy for membrane proteins,**
- **83% accuracy for extracellular proteins, and**
- **83% for anchored membrane proteins**

on the 200 out-of-sample proteins (off-line) of Cedano et al. 1997.

- **The evolved classifiers employed a large variety of types of computation that were available in genetic programming, but not available in more constrained computational techniques.**