Artificial Intelligence in Medicine (2005) **35**, 227–241

ELSEVIER

ARTIFICIAL INTELLIGENCE IN MEDICINE

http://www.intl.elsevierhealth.com/journals/aiim

Genetic algorithm for analysis of mutations in Parkinson's disease

Rafal Smigrodzki [a], Ben Goertzel [b], Cassio Pennachin [b], Lucio Coelho [b], Francisco Prosdocimi [b], W. Davis Parker Jr. [a,*]

[a] *Department of Neurology, University of Virginia School of Medicine, Charlottesville, VA 22908, USA*
[b] *Biomind LLC, 11160 Veirs Mill Rd., L-15, Suite 161, Silver Spring, MD 20902, USA*

Received 20 July 2004; received in revised form 4 November 2004; accepted 12 November 2004

KEYWORDS
Genetic algorithm;
Mitochondrial DNA;
Pattern recognition;
Parkinson's disease

Summary

Objective: Mitochondrial genetics has unique features that impede analysis of the biological significance of mitochondrial mutations. Simple searches for differences in total mutational load between normal and pathological samples have been frequently unrewarding, raising the possibility that more complex patterns of mutations may be responsible for some conditions. We explore this possibility in the context of Parkinson's disease (PD).
Methods and materials: We report the development of a modified genetic algorithm suited for detection of biologically meaningful patterns of mitochondrial mutations. The algorithm is applied to a database of mutations derived from biological samples, and verified by the use of shuffled data, and repeated leave-one-out testing.
Results: It is possible to derive, from a very small sample, multiple accurate classifier functions that correlate with biological features. The methodology is validated statistically through experiments with fabricated data.
Conclusion: This algorithm might be generally applicable to conditions where interactions among multiple mitochondrial DNA mutations are important. The patterns embodied in the classifier functions obtained should be the subject of further experimental study.

1. Introduction

In the post-genomics era of bioinformatics, there is a crucial need for computational analysis tools that can help researchers understand properties of genomics data, be it sequence, expression, or mutation

* Corresponding author. Tel.: +1 434 924 8378; fax: +1 434 982 1776.
E-mail address: dp8m@virginia.edu (W.D. Parker Jr.).

0933-3657/$ – see front matter
doi:10.1016/j.artmed.2004.11.006

data. Often, important phenotype features cannot be easily traced to mutations or the expression patterns of a single gene, and interactions between multiple genes need to be modeled and studied. Mutation patterns linked to genetic diseases, for instance, can be complex and require advanced analytic tools for their identification and understanding. This need is especially pronounced for mitochondrial conditions, where the complexity of mutation and inheritance patterns makes classical genetic analysis difficult.

Mitochondria are intracellular organelles involved in oxidative metabolism and other cellular processes, distinguished by the presence of the mitochondrial genome (mtDNA) that is inherited independently from the nuclear genome, codes for 13 peptides forming a part of the electron transport chain (ETC), and has unique properties, which present problems for the analysis of mtDNA sequence data. Unlike nuclear genes that are present in only two copies in each cell, the mitochondrial genome is present in every cell in many copies ranging from a few in platelets to as many as 50,000 in a neuron. Furthermore, the mitochondrial genotype may be non-uniform in a single cell. Rather, there are complex mixtures of mitochondrial genotypes, a condition termed heteroplasmy. A large number of pathological conditions have been correlated to heteroplasmic mutations, with mutated mtDNA fractions sufficient for clinical symptoms ranging from 30 to 90% [1,2]. In addition to this high-level heteroplasmy, extensive sequence studies of multiple copies of mitochondrial genes from single individuals have shown that low-level heteroplasmy (1—5%, microheteroplasmy) is extremely common in humans in all mitochondrial genes [3—5]. In elderly humans, the total compound mutation burden is estimated to be 0.3—0.6 amino acid-changing mutations in each set of the seven genes encoding ETC complex I peptides, a fraction comparable to the mutation burden in classical high-level heteroplasmic mitochondrial disease. This mutation burden is present in all groups of elderly individuals examined so far, including normal controls, and appears to increase with age in most tissues [4]. Given this complex situation, the question arises whether global or focal (gene segment-specific) levels of microheteroplasmy may be correlated with pathophysiological features, and how to detect such correlations.

We report the development and application of an analytic approach involving a combination of evolutionary conservation scoring, amino acid dissimilarity matrices and a genetic algorithm to the detection of patterns in the frequency of microheteroplasmic mutations at particular segments of the mitochondrial genome. A special-purpose modification of the simple genetic algorithm was developed for the analytical work [6,7]. This genetic algorithm, described in Section 2.2, is able to cope well with very sparse rules covering only a small subset of the space of possible patterns. Since the detection of microheteroplasmy is technically challenging, it is important to be able to detect patterns in small numbers of samples. As a test case, we applied this algorithm to a mutation database derived from Parkinson's disease (PD) patients allowing construction of classifier functions correlating mutations in some segments of mtDNA with disease status, using only 12 samples.

2. Materials and methods

2.1. Data collection

Complete sequences for all 7 ND genes (NADH: ubiquinone oxidoreductase, or mitochondrial complex I) in mitochondrial DNA were obtained for 12 brain samples—6 with idiopathic PD and 6 age-matched controls. The choice of PD as the test case was dictated by reports correlating complex I dysfunction and PD. Depressed activity of complex I, which contains 7 mitochondrial-encoded proteins (ND genes) in addition to approximately 40 nuclear-encoded proteins, is found in sporadic PD [8]. The loss of complex I activity in idiopathic PD appears to arise from mtDNA as evidenced by experiments in which patient mtDNA is expressed in cultured cells depleted of endogenous mtDNA, with subsequent emergence of a specific mitochondrial biochemical defect [9—11].

Sample collection, DNA purification, cloning, sequencing, and mutation validation were performed as described [5]. On average, each gene was completely sequenced 94 times in each sample. A total of 659 different microheteroplasmic mutations were detected in complex I genes, at levels of 1—2% of the total number of mtDNA genomes, including 411 amino acid changing substitutions and 118 frameshifts, which form the input to the algorithm. As an example, amino acid-changing substitutions found in all 12 samples in the gene ND5 are listed in Table 1. The wild-type amino acid (in the standard single-letter amino acid code [12]), the location of the mutation on the mitochondrial genome (using the standard numbering of the consensus human mtDNA [13]) and the amino acid resulting from the mutation are given.

Only a few mutations are found in more than one sample [5]. High-level heteroplasmic and homoplasmic mutations (mutations present in 30—100% of the

Table 1 Amino-acid changing mutations found in gene ND5

Patient	Mutations
Ct1	M108T T284A A408V N444K A501G F528L R535G L553V P563R I596M L601V
Ct2	A458T L555M
Ct3	T43A D58G N175S L227F I257V I395M I468M A492T H534Y T598S
Ct4	I301M G376C A399T A458T T544A
Ct5	A168G G222R E238K L246V K300E T481K T500K
Ct6	C279W F287L L293V A475G L488V A490G A492G D503E S545T
PD1	K28E A41G I130L A134G E145D T306S L310V F334L L350H A458T T544A
PD2	L57V L260I S343T S372T Q568L L599P
PD3	E102G N136D E145G N175K L266V F385L S461I N505T T536M L550V
PD4	E145V A225G I454F A475G A490G A492G G497A C518G F54I
PD5	T133A P265T A399E S423N R425P P448H A490G H534L L540M T603K
PD6	L125P R161L L191P H230Q R262H T294M I317T M341T R357Q A399G T556A

genomes in a patient) are excluded from the analysis, since a very large number of studies were performed on them finding no convincing correlations with pathology (for a brief listing, see [5]). The average number of mutations found was 59.3 mutations per million bases sequenced, which corresponds to 32% of mitochondrial genomes having at least one amino acid-changing mutation in complex I genes.

2.2. The Evolutionary Classification System

This section describes the analytical method employed in this study. As described below, this same general approach method was used in two ways, one simple and one more complex involving consideration of specific amino acid changes and protein conservation.

The crux of our analytical approach is a special evolutionary learning technique, inspired by traditional genetic algorithms (GA) [6]. Genetic algorithms are a loosely biologically inspired optimization technique, in which a population of candidate solutions is *evolved* through simulated evolutionary dynamics. The candidate solutions are described by a string which functions as a ''genome''—the genome is interpreted as a solution for the problem at hand by a user-defined function, which assigns a *fitness value* to each candidate solution. The best candidates in the population are then selected for reproduction (these are called *parents*), and new solutions (called *offspring*) are generated by the genetic operations of crossover and mutation, acting on the genome of the selected parents. Typically, two parents are combined to generate one offspring. Their genomes are crossed and then mutated. Once the offspring are as numerous as the original population, they are evaluated for fitness. The process iterates for a fixed number of generations, or until a desired fitness level is reached.

The use of genetic algorithms in this problem is justified by two properties: GAs are able to learn good solutions to classification problems from a small sample, as is the current case, and their performance is not highly dependent on the nature of the problem at hand, or the encoding of the inputs, as with alternate machine learning techniques, due to the separation between solution encoding (the *genome*) and its evaluation (the *fitness function*). This separation allowed us to quickly develop specialized encoding and fitness functions specifically for the problem at hand.

In our representation, we utilize a special alphabet for each gene, which stores a tuple, instead of the usual binary, discrete or float-valued genes. Also, our individuals have variable length, and a special crossover operator. We used this modified GA to evolve *classifiers*—i.e., classification rules that indicate whether a given set of mitochondrial genes comes from a person with Parkinson's or not.

We refer to this learning system as the Evolutionary Classification System (ECS). Individuals in ECS are sets of gene segment definitions. In this context, a gene segment definition s is a tuple, $s = (g; b; l; m)$, where:

- g is a gene in G, the set of genes being considered;
- b is the start point (in terms of amino acids, not bases) of the segment in g;
- l is the length of the segment in amino acids, which has to be no greater than a given maximum L;
- $m \in \{\text{Control; PD}\}$ is a label assigning that segment as relevant to Parkinson's or Control.

In our case, G is the set of seven mitochondrial genes: ND1, ND2, ND3, ND4, ND4L, ND5, and ND6.

Each individual in the first generation is created as follows:

1. Let $I \leftarrow \{\}$ be the individual.
2. For each gene $g \in G$:
   2.1. Repeat length(g) $\div$ L times:
   2.2. Generate a segment $s = (g; b; l; m)$ in g by randomly choosing a length $1 \leq l \leq L$, a start point $1 \leq b \leq \text{length}(g) - l + 1$ (where length(g) is the length of the gene g in amino acids) and a label $m \in$ {Control; PD}.
   3. If s does not overlap any segment in I, then $I \leftarrow I \cup \{s\}$.

Specialized crossover and mutation operators perform the creation of individuals in the next generations. The crossover operator works as follows:

1. Let $I_r \leftarrow \{\}$ be the new individual being created by crossover.
2. For each parent P, do
   2.1. Let $1 \leq n \leq \text{length}(P)$ (where length(P) is the number of segments of P) be the randomly chosen number of segments from P to be added.
   2.2. While $n > 0$, do:
      2.2.1. Randomly choose a segment $s_r \in P$.
      2.2.2. If s_r does not overlap any segment in I_r, then $I_r \leftarrow I_r \cup \{s_r\}$.
      2.2.3. $n \leftarrow n - 1$;

Finally, mutation is accomplished in the following way: for each newly generated individual I_r, a randomly generated segment $s = (g_s; b_s; l_s; m_s)$ (where g_s is a randomly chosen gene from G, $1 \leq b_s \leq \text{length}(g_s) - l_s + 1$ is a randomly chosen start point, $1 \leq l_s \leq L$ is a randomly chosen length, and m_s is a random label in {Control; PD} is added to I_r if it does not overlap any already-present segment in I_r.

The individuals thus generated are interpreted as segment maps which are used as classifiers. The classification function is used to determine the fitness of each individual. Specific classification and fitness functions are given in Sections 2.3 and 2.4.

All the results in the next section were based on 1000 runs of ECS, always using populations of 100 individuals evolved for 100 generations. ECS is a generational GA with elitism, i.e., it selects parents from the current population to generate a new population which replaces the original one; however, it will always preserve the best element in the original population, so only 99 new elements are generated for the new population, and the best parent in the original population survives. The mutation rate was fixed in 0.01, and the crossover rate was set to 1.0. Parent selection was performed by a tournament of size 2 for the generation of each new element, i.e., two random individuals are chosen from the current population, and the one with best fitness is allowed to reproduce; this process is repeated twice for each new individual being generated. At the end of each evolution, the best-of-the-run individuals are selected as the resulting classifier. Thus, 1000 classifiers are produced. The maximum length allowed for a gene segment definition was set to $L = 7$, in order to privilege small segments and thus facilitate possible future experiments with site-directed mutagenesis. Also, in order to obtain smaller classifiers which are easier to understand, we used the reward parameter set to $r = 0.1$. This small evolutionary pressure led to smaller classifiers, but it did not impoverish the results.

2.3. Learning simple classifiers

This section describes the initial results obtained with the methodology described in the previous sections. In the first round of experiments, we used simple classifiers, as described in Section 2.2. In this scheme, all mutations are considered to be equal (all the amino acid substitutions, for example, are considered to have the same potential impact on the protein produced, and no distinction were made between mutations in conserved domains or outside such domains).

The individual creation and mutation process described in Section 2.2 was simplified slightly for the first round of experiments. All gene segment definitions in all individuals only contained segments labeled as PD. The classification function used to evaluate the mutation map defined by an individual was quite simple as well: a patient is classified as PD if it has at least one mutation that hits one of the segment definitions in the classifier, and as Control otherwise.

The fitness function was a simple hit-based function, giving us a discrete fitness landscape, i.e., the search space defined by the fitness function was always discrete and therefore, smaller, making classifier learning less computationally intensive.

2.4. Learning classifiers with mutation weight

Generally, it is biologically suspect to give equal importance to all mutations, as is done in the methodology used above. This section describes another round of computational experiments, in which the definition of classification rules was made more sophisticated. This time, we allowed gene segment

definitions to be labeled as Control as well as PD. Also, we weighted mutations in three ways. We used conservation patterns to identify mutations that occurred on evolutionarily conserved regions. We also observed whether a mutation resulted in a drastic change in the neighborhood of the new amino acid, based on public data about which amino acids are frequently in the neighborhood of others. Finally, a new scoring for frameshifts was used.

2.4.1. Conservation patterns

Conservation patterns use protein sequences for homologous proteins from different species to identify regions that are present in most of those proteins. These regions are assumed to be conserved throughout the evolution process, and an argument can be made that they are important for protein function. Therefore, substitution mutations in evolutionarily conserved regions are more likely to have an effect on protein function than mutations in evolutionarily unstable regions.

We used SIFT [14] which uses sequence similarity information from BLAST [15] to compute the diversity at each position of a target protein. The protein sequences were obtained from SWISS-PROT [16]. Given a list of substitution mutation position and descriptions, SIFT assigned to each one the label of tolerant or intolerant. Intolerant mutations were given higher weights.

2.4.2. Amino acid dissimilarities

Another indication of the potential impact of a substitution mutation to protein function is the neighborhood amino acid dissimilarity index of Xia and Xie [17]. The Xia and Xie amino acid dissimilarity matrix was computed to reflect the thermodynamic cost of replacing one amino acid with another, considering the impact on its neighborhood. Certain changes to the neighborhood are more likely to affect protein function and are, therefore, given higher weight.

2.4.3. Frameshifts

The impact of frameshift mutations on protein function is expected to be much higher than the impact of substitution mutations. To handle this higher impact, we defined, for each gene, a point, called the *P* point. Any frameshift mutations before the *P* point were considered severe enough to render the protein non-functional. The *P* point for each gene was chosen by visual inspection of the results of multiple alignments between the target protein and similar proteins obtained from the sequences in SWISS-PROT. The *P* point was chosen to be the last amino acid in the last conserved domain in the target protein. Table 2 shows the protein length and the *P* point for the seven genes.

Table 2 *P* points and protein sizes

Gene	*P* point	Size
ND1	286	318
ND2	295	347
ND3	80	115
ND4	459	459
ND4L	75	98
ND5	545	603
ND6	150	174

2.4.4. Classification and fitness functions

In our revised approach, the classification formula incorporated a weight component, for each mutation, which incorporated the biological knowledge described in the previous subsections.

The weight given to substitutions in the GA was calculated as the sum of the value on the Xie and Xia neighborhood substitution matrix [17] plus a value for tolerant or intolerant positions, according to the output by SIFT. Mutations in conserved positions (SIFT intolerant) added 10 points to the dissimilarity weight, while mutations in non-conserved ones (SIFT tolerated) added one single point. Given the range of values in the neighborhood matrix (from 0.47 for the substitutions *L*—*I* and *M*—*L* to 5.38 for *W*—*D* mutation), this means that the conservation profile dominates the weights for substitution mutations.

For a substitution mutation in which amino acid *a1* is replaced by amino acid *a2*, the weight $w_S(a1;a2)$ is given by:

$$w_S(a1;a2) = \text{Neighborhood}(a1;a2) + t \tag{2.1}$$

where *t* is the tolerant or intolerant weighting factor (1 or 10).

Frameshift mutations had a maximum weight, w_{MAX}. Several values were experimented with for this limit. All frameshift mutations before or at the *P* point were given the maximum weight. Mutations after the *P* point were weighed linearly, decreasing from w_{MAX} at the *P* point to 1 at the last amino acid.

For a frameshift mutation at position p_m, in a protein of size *S*, the weight w_F is given by:

$$w_F = \begin{cases} w_{MAX}, & \text{if } p_m \le P \\ \left(1 - \dfrac{p_m - P}{S - P}\right) w_{MAX}, & \text{otherwise} \end{cases} \tag{2.2}$$

The classification function was redefined as:

$$C(I,h) = \begin{cases} \dfrac{W_{PD}}{W_{Control} + W_{PD}}, & \text{if } W_{Control} + W_{PD} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.3}$$

Table 3 Examples of classifiers evolved by ECS

```
Classifier:
      {(ND1,11,17,PD), (ND2,2,8,PD), (ND5,130,132,PD)}
Out of Sample Patient:
      BR14, PD
Match:
      BR14: mutation in ND5, location 130, weight 0.47
Result:
      PD, correct

Classifier:
      {(ND1,16,18,PD), (ND2,4,8,PD),
      (ND4,23,26,Control), (ND5,39,41,PD)}
Out of Sample Patient:
      BRF174, PD
Match:
      BRF174: frameshift in ND1, location 17, weight 100.0
      BRF174: mutation in ND2, location 5 ,weight 1.23
Result:
      PD, correct

Classifier:
      {(ND1,104,111,PD), (ND2,1,6,PD), (ND5,129,134,PD)}
Out of Sample Patient:
      AG760, Control
Match:
      No matching mutations
Result:
      Control, correct
```

where I is the individual being used as a classifier; $h \in H$ is the patient being classified (diagnosed), and H is the set of patients being used for training; W_{PD} is the sum of the weights of the mutations of h that fall inside segments of I labeled as "PD"; $W_{Control}$ is the sum of the weights of the mutations of h that fall inside segments of I labeled as "Control".

Given this classification function, the fitness function is defined as:

$$F(I,H) = \frac{1}{1+\text{error}(I,H)} + \frac{r}{1+\text{length}(I)}, \tag{2.4}$$

where

$$\text{error}(I,H) = \sum_{h\in H}^{H} (\text{expected}(h) - C(I,h))$$

$$\text{expected}(h) = \begin{cases} 0, & \text{if } h \text{ is a Control patient} \\ 1, & \text{if } h \text{ is a PD patient} \end{cases}$$

r is a reward parameter favoring GA-individuals composed by smaller numbers of gene segments.That is, the fitness function is approximately the inverse of the sum of the errors of the classifier tested for all patients in the training set, plus a reward term that favors smaller GA-individuals.

3. Results

3.1. Results for the simplified algorithm

As mentioned in Section 2.2, the ECS was run 1000 times, to produce 1000 classifiers. However, the ECS runs were repeated 12 times as part of method validation through leave-one-out testing (see Section 4.1), generating the grand total of 12,000 classifiers. Each classifier consists of a map of segments within the ND genes, covering various combinations of codons, assigned the label "PD". The usual number of segments per classifier was 3–5 (see Table 3 for examples). The total number of codon positions on segments in all 12,000 segments classifiers was 61,618, with most segments clustering in some areas of the ND genes, and overlapping at many codons. The 10 codons, which were used most frequently to make the classification of PD patients, are listed in Table 4. The most important codons are concentrated in genes ND2, ND4, and

Table 4 Genes and positions most frequently used to identify PD on non-weighted algorithm

Gene	Position	Percentage in top 10 codons
ND5	145	30.89
ND4	180	9.84
ND5	146	6.76
ND5	148	5.44
ND2	272	5.11
ND4	236	4.68
ND2	273	4.26
ND2	270	4.16
ND4L	49	3.17
ND4	183	3.12

The percentages given are the relative frequencies of codon usage in classification, among the 20 most frequently used codons.

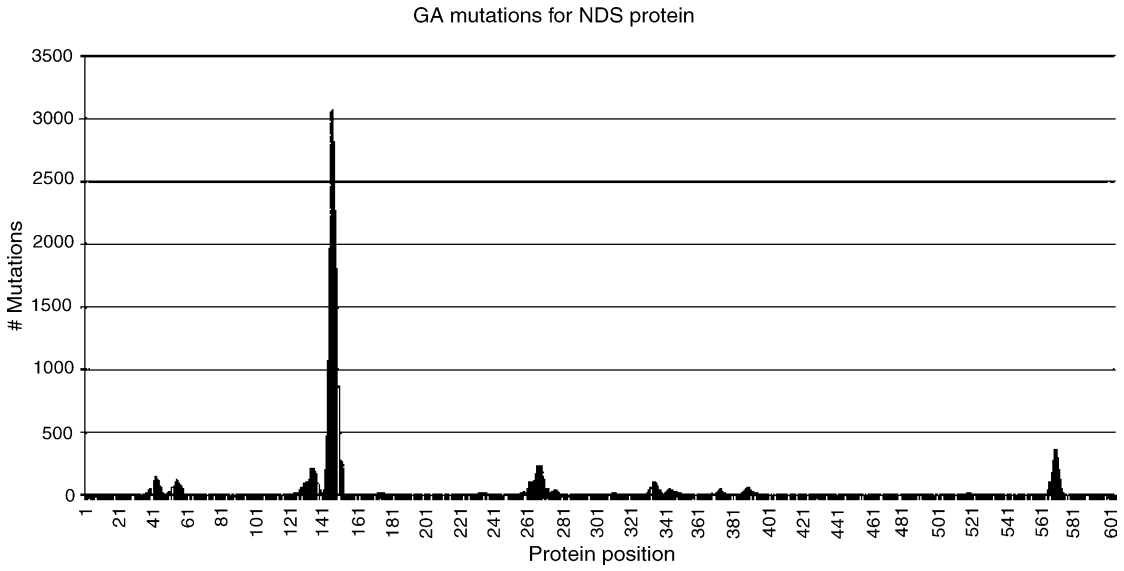

Figure 1 Positions found on PD related sequences by ECS for ND5 (simplified algorithm).

ND5. Fig. 1 shows the absolute number of times each codon position was a part of a PD-segment, in all 12,000 classifiers, in gene ND5. Thus, codon 145 in ND5 was used by 3063 out of 12,000 classifiers. The *y*-axis provides a measure of the relative importance of each codon for the classification of samples, although it is not a direct expression of the statistical significance of the codon.

3.2. Results for algorithm with weights

We ran experiments with various values for the maximum weight for frameshift mutations before or at the *P* point, the w_{MAX} parameter. One hundred ECS runs were performed for each of the following values: 25, 50, 100, 200, and 300. The best results were obtained with w_{MAX} set to 100 (data not shown), and that value was used for the 1000 runs that produced the following results. Again, 12,000 classifiers were produced in the course of leave-one-out testing.

Table 5 Genes and positions most frequently used to identify PD on weighted algorithm

Gene	Position	Percentage in top 10 codons
ND5	145	17.33
ND2	5	11.60
ND1	314	8.53
ND5	266	7.53
ND2	187	6.69
ND2	6	6.18
ND5	146	5.09
ND5	268	3.78
ND5	265	3.38
ND2	188	3.29

The percentages given are the relative frequencies of codon usage in classification, among the 20 most frequently used codons.

Table 5 is similar to Table 4, and it shows the 10 codons that were most frequently used in classification. We can see that the top position remains unchanged, but the others vary. Based on this table, the genes ND1, ND2, and ND5, seem to be the most relevant for a mutation-based classification of PD.

Figs. 2 and 3 show the absolute number of times each codon in genes ND5 and ND6 (for comparison) was used to classify a patient as PD, in all 12,000 classifiers. Although the total mutation burdens over the whole complex I do not differ between PD and controls, the mutation burden in the peaks on graphs 3—5 is much higher in PD than in control. For example, the portions of peaks with *y*-axis values that are higher than 500 in all 7 complex I genes, contain 29 mutations in the 6 PD datasets, versus only 2 mutations in controls.

An analysis of the mutation distributions shows that the classifiers obtained using weighted mutations are slightly different from those obtained using the previous, simpler methodology. Their usage of mutations is not as concentrated as is the case when mutation weights are ignored.

4. Validation of the methodology

Due to the innovative nature of our methodology and the relatively small size of our dataset, we have paid particular attention to issues of validation. In this section, we will describe the various computa-

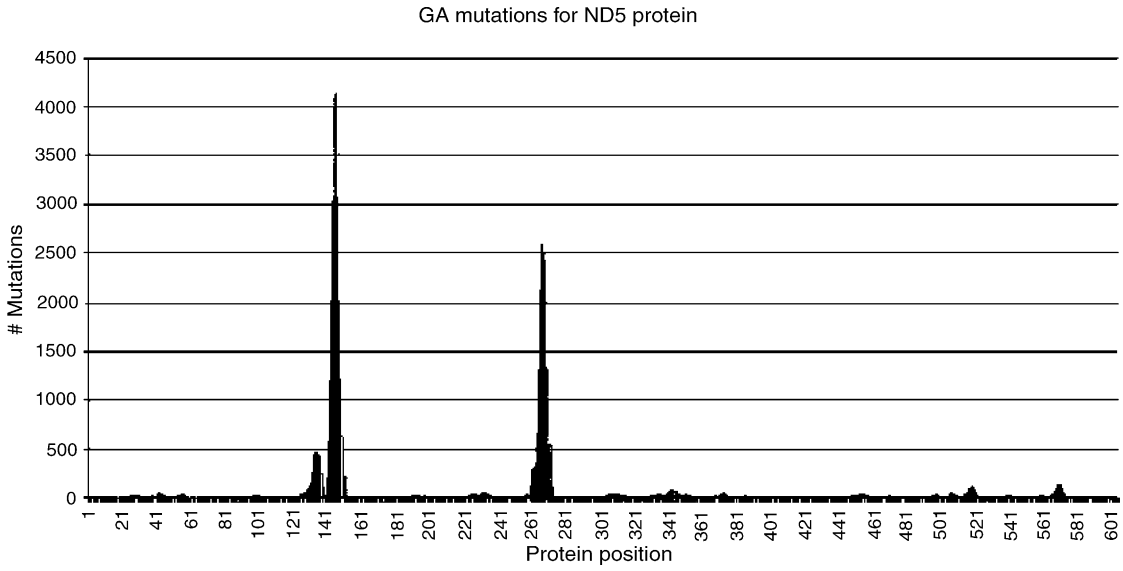

Figure 2 Positions found on PD related sequences by ECS for ND5 (weighted algorithm).

tional experiments we have run to ensure the validity of results. These include:

- cross-validation, using leave-one-out and leave-two-out testing;
- analysis of shuffled data (datasets produced by random shuffling of patient labels);
- analysis of data similar in form to the real data using fabricated mutations;
- analysis of random mock data using the same methodology.

The overall conclusion of these experiments is that the results obtained appear to be genuine and significant. The cross-validation experiments provide a quantitative assessment of significance, which is valid even in this small sample; the shuffled experiments can also be used for quantitative validation; additionally, the shuffled and randomized data experiments provide qualitative indications that the patterns found by our methodology are real rather than artefactual.

4.1. Cross-validation results

The primary validation technique used in this research was cross-validation [18]. In *k-fold cross-validation*, one divides the *n* data sets into *k* subsets

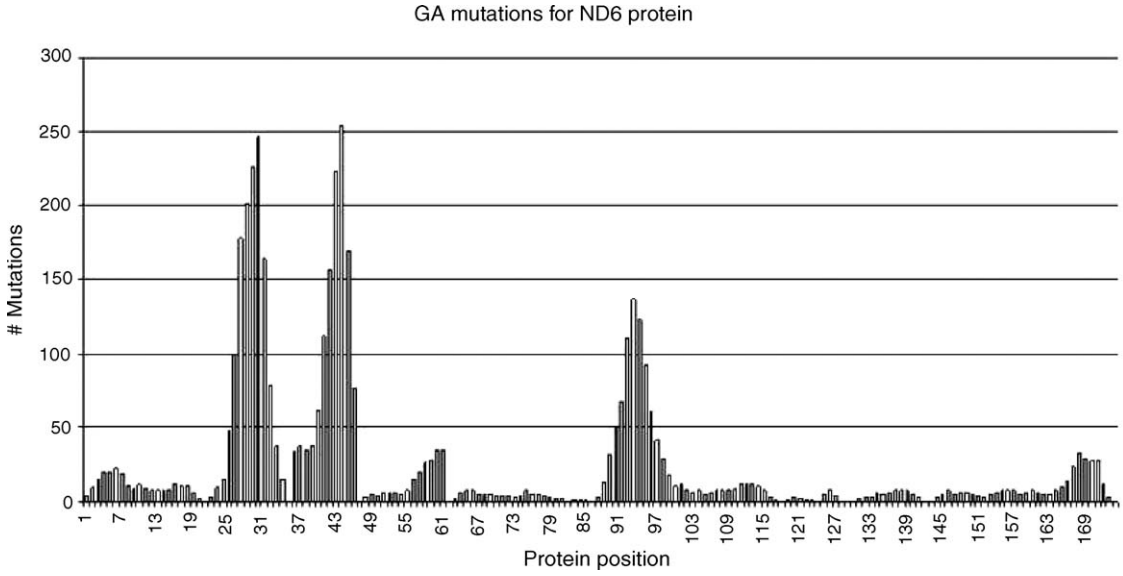

Figure 3 Positions found on PD related sequences by ECS for ND6 (weighted algorithm). It can be seen that many fewer mutations were found in this gene than in ND1, ND2, and ND5.

of (approximately) equal size. One then runs the model-learning k times, each time leaving out one of the subsets from training, but using only the omitted subset to compute the classification accuracy. If k equals the sample size, this is called *leave-one-out* cross-validation. *Leave-v-out* is a more elaborate and expensive version of cross-validation, which requires leaving out all possible subsets of v cases. For example, on our dataset with an $n = 12$, leave-one-out validation requires doing model-learning 12 times, and leave-two-out validation involves doing model-learning 66 times. In a stochastic learning process such as the ECS, the trials may be repeated, in our case 1000 times, producing 12,000 and 66,000 classifiers, respectively.

Note that cross-validation is quite different from the *split-sample* or *hold-out* method that is commonly used for validation. In the split-sample method, only a single subset (the validation set) is used to estimate accuracy, instead of k different subsets; i.e., there is no crossing. In the case of small datasets, cross-validation is markedly superior, as has been repeatedly demonstrated, e.g. by [19]. The theoretical properties of cross-validation have been explored extensively; see, for instance, [20]. For these reasons, split-sample experiments were not performed.

In our leave-one-out testing, all of the 12,000 classifiers correctly identified the left-out sample, both in the weighted and non-weighted approaches. Since there were 12 fully independent sets of 1000 classifiers each (derived for each of the 12 possible combinations of 11 learning datasets and the 12th verification dataset), the lower bound of the statistical significance of the results may be calculated as follows.

First, the outcome of a single leave-one-out test (1000 ECS runs on one combination of learning and verification datasets) is scored as correct when all of the classifiers (1000) correctly identify the left-out dataset, and as incorrect if one or more classifiers mislabel the left-out dataset. This scoring accounts for the interdependence of the individual classifiers within a leave-one-out test, and allows transforming the test output into a binary format (correct versus incorrect). Lower thresholds for scoring the test (e.g. correct if 90% of classifiers are correct) may be used but they do not fully account for the interdependence of classifiers, which are merely re-runs of the same stochastic algorithm on the same learning data, and may exhibit significant convergence without finding true patterns.

Subsequently, the statistical significance is calculated as the probability of twelve independent test runs with correct binary outcomes occurring randomly in a sample with equal number of PD and controls, and it equals $P = 1/2^{12} = 0.00024$. Since the ECS classifiers within each leave-one-out test are not fully convergent (the classifiers are not identical in each batch of 1000), this number is a lower bound estimate of the likelihood that PD-correlated patterns exist in our sample of mutations. Calculation of the exact statistical significance is beyond the scope of this work.

In order to obtain further assurance of the validity of our method, we also ran tests using leave-two-

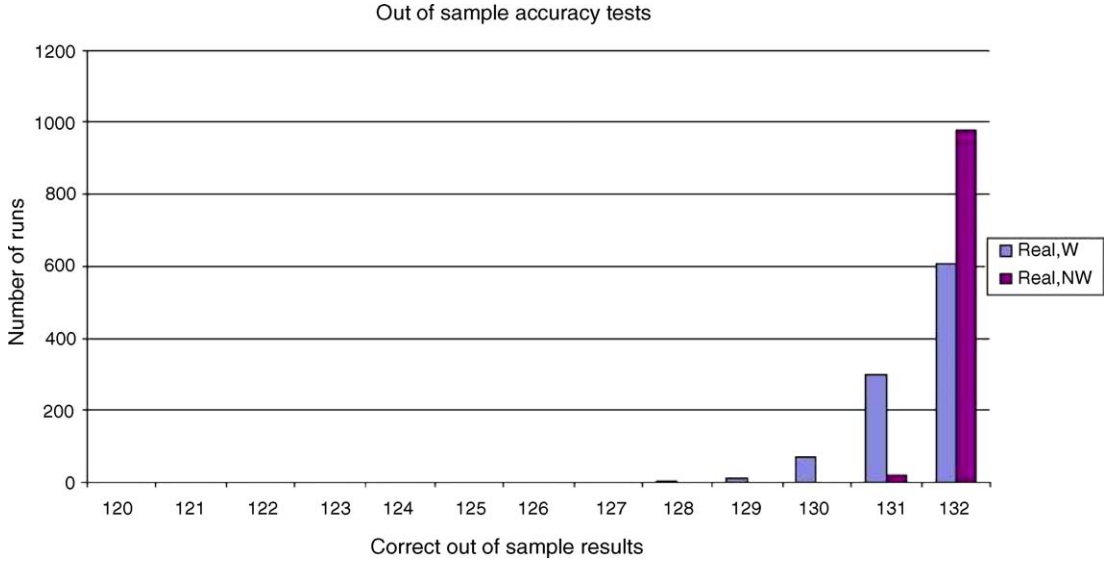

Figure 4 Correct out of sample results for the leave-two-out tests. The classification was performed 132 times, for 66 combinations each consisting of 10 learning datasets and 2 verification datasets. The worst result classifies correctly 128/132 datasets and the better result classifies all of them correctly (980/1000 times for non-weighted algorithm and 610/1000 times for the weighted one).

Table 6 The mean accuracies and standard deviations of classifiers from shuffled data and actual data experiments

Experiment	Normal data		Shuffled data		z-Test
	Mean	S.D.	Mean	S.D.	
Non-weighted	1.0	0.0	0.968	0.053	19.317
Weighted	1.0	0.0	0.937	0.066	30.001

Table 7 The mean and standard deviation of the number of segments in classifiers from shuffled vs. actual data experiments

Experiment	Normal data		Shuffled data		z-Test
	Mean	S.D.	Mean	S.D.	
Non-weighted	2.017	0.128	2.667	0.879	8.103
Weighted	3.092	0.365	3.933	1.487	6.199

out cross-validation. These results were similar to those obtained using leave-one-out. The accuracies obtained are shown in Fig. 4. The ECS was run 66,000 times, 1000 times per each combination of 10 learning datasets and 2 verification datasets. This resulted in 132,000 assignments of PD or control label to the verification datasets ($2 \times 66 \times 1000$). Over 99% of the assignments were correct.

4.2. Analysis on shuffled data

As another way of exploring the validity of the results, we also ran the same analytic procedure with shuffled data. We generated new mutation files for 12 hypothetical patients, using the original mutation data but randomly re-assigning the PD and control labels among patients. Results of the shuffled data runs support the validity of the ECS classifier-learning methodology—although, a proper analysis of the shuffled data results turns out to require some care, for reasons to be elucidated in the following discussion.

The ECS was run in the leave-one-out paradigm 12,000 times, and the fractions of classifiers correctly labeling the left-out dataset are given in Table 6, for both the weighted and non-weighted experiments, compared to the results from actual data. The z-tests on that table imply that the solutions for the original data contain significantly more information than solutions for the shuffled data, which is as expected. However, the classification accuracies obtained on the shuffled data are also reasonably good, a point which seems surprising until one analyzes the matter in detail, as we will do in the following section.

Table 7 shows the mean and variance for the distribution of classifier sizes (in terms of number of segments) for normal and shuffled data, for both the weighted and non-weighted experiments. What the z-tests here show is that the classifiers found for the shuffled data are consistently larger, involving significantly more segments than the classifiers found in the actual data. This is a good illustration of the maxim of "minimum description length" modeling, which holds, in essence, that more compact models are more likely to possess true statistical validity [21].

In conclusion, these shuffled-data results show that the results obtained on the data are highly significantly better than results obtained on shuffled versions of the data, involving significantly higher accuracies and significantly smaller models. However, the accuracies on the shuffled data are higher than one might initially expect. There are two possible explanations for this: a flaw in the analytic methodology or richness of the dataset, which means that for many random partitions, reasonably decent classifiers may be learned.

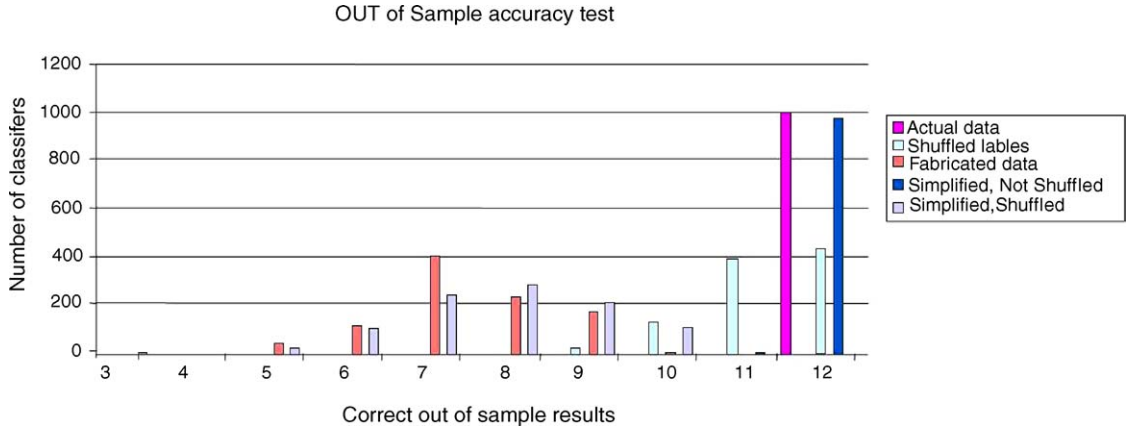

Figure 5 Accuracy of the ECS depending on the input. The results for actual data analyzed with the weighted algorithm compared to shuffled data, fabricated data, and simplified fabricated data.

Table 8 Examples of data shufflings and the strategies evolved by ECS on shuffled data

Patient	Original label	Shuffled label	ECS classification
(a)[a]			
BR14	PD	Control	PD
BR18	PD	PD	PD
BRF168	PD	Control	PD
BRF172	PD	PD	PD
BRF174	PD	PD	PD
AG720	Control	Control	Control
AG760	Control	PD	Control
AG761	Control	Control	Control
BR15	Control	Control	Control
BR16	Control	PD	Control
BR17	Control	PD	Control
(b)[b]			
BR14	PD	Control	Control
BR18	PD	Control	Control
BRF168	PD	Control	Control
BRF172	PD	PD	Control
BRF174	PD	PD	Control
AG720	Control	Control	Control
AG760	Control	Control	Control
AG761	Control	Control	Control
BR15	Control	Control	Control
BR16	Control	Control	Control
BR17	Control	Control	Control
(c)[c]			
BR14	PD	Control	Control
BR18	PD	Control	Control
BRF168	PD	PD	Control
BRF172	PD	Control	Control
BRF174	PD	Control	Control
AG720	Control	Control	PD
AG760	Control	PD	PD
AG761	Control	Control	PD
BR15	Control	PD	PD
BR16	Control	PD	PD
BR17	Control	PD	PD
(d)[d]			
BR14	PD	PD	PD
BR18	PD	Control	PD
BRF168	PD	PD	PD
BRF172	PD	Control	PD
BRF174	PD	PD	PD
AG720	Control	PD	PD
AG760	Control	PD	PD
AG761	Control	PD	PD
BR15	Control	PD	PD
BR16	Control	PD	PD
BR17	Control	Control	PD

[a] For this shuffling, it was slightly profitable to preserve the original labels. Note that assigning PD (1.0) labels to all controls would have been equally fit. Reverting the original labels, or assigning Control to all patients would have been worse.

[b] Here, the shuffling preserved all controls and shifted most PDs. So the most profitable strategy was to classify all patients as controls.

[c] Here, the shuffling inverted most labels, making it most profitable to actually learn patterns from the data, but assign them reverted labels.

[d] Finally, a case in which most PD labels were preserved, and most Controls were shifted. So it was profitable to assign a PD label to everyone.

The correct explanation is the latter one, as demonstrated by further computational experiments, described in the following sections. These further experiments are informative in terms of the validity of the results, and also provide insight into the nature of the ECS analytical methodology.

Fig. 5 shows out-of-sample accuracies for the real and shuffled data tests, and also for the subsequent tests to be described in following sections. All examples use leave-one-out cross validation.

4.3. Analysis of fabricated mutation data

In order to further explore the issues raised by the shuffled data experiments, we ran a computational experiment referred to as the ''fabricated mutation data'' experiment, in which we replaced *all* the data with data in exactly the same format, but involving different contents. In this fabricated dataset, all Control patients had the same set of mutations, in gene ND1. All PD patients had a different, common set of mutations, in gene ND2. No other mutations were present in the data. Thus, any mutation in ND1 identified a Control patient, and any mutation in ND2 identified a PD patient.

Not surprisingly, results on this fabricated dataset were not nearly as good as the results obtained with the original data and shuffled labels. In this rather poor data, the results are poor as well. However, this experiment does highlight some interesting phenomena. As Fig. 5 shows, even for this dataset, the results are not centered on 50% accuracy.

The nature of the genetic algorithm at the heart of the ECS learning methodology provides an explanation. Given the real patterns in the data, there are four strategies for ECS:

1. evolve a model that reflects the real patterns;
2. evolve a model that reverts the real patterns;
3. evolve a model that classifies all patients as PD;
4. evolve a model that classifies all patients as Control.

It seems to be the case that, for most partitionings of the labels, one of these strategies will give performance better than 50%, on average, over the 12 different leave-one-out runs done on that shuffling. Which strategy does best will depend upon the shuffling in question. Evidence for this explanation is provided by the observation that, in 92 out of 100 runs, all 12 of the models for the run *converge on only one of these four strategies*. Table 8(a–d) show four runs, each of these runs corresponding to a situation in which one of the four patterns was profitable.

Given the data, and the format of our classifiers (a classifier is a set of gene segment definitions with labels) all four strategies are possible. Hence, from simple patient label shuffling, one can expect imperfect, but better than random results, as the data is exploitable even when shuffled.

4.4. Analysis of simplified fabricated data

Finally, we ran a computational experiment involving a completely different sort of data, but using the same GA, and the same leave-one-out model learning and evaluation code.

These new data were entirely fabricated and were significantly simpler than the real data, their shuffled varieties, or the fabricated-mutation data considered above. The data consisted of a binary string of length 100 for each patient. Patients were considered as PDs when they had 51 or more bits set to 1. A random dataset for this experiment, with seven PD patients, and five Controls, is shown in Table 9.

The classifier was another 100-bit string, which would represent a linear partition in this one-dimensional space: the number of set bits would be a threshold. Patients with more 1 s than the threshold would be considered PDs, and vice versa.

Note that learning this threshold by encoding it in the number of set bits makes the problem much harder for the GA than it could be with a more sensible encoding, because equally fit individuals are less likely to share the same schema [6]. This was by design, because we wanted the GA to have to work at the limit to its capabilities to find perfect classifiers even in the non-shuffled data. The results from this experiment are shown in Table 10. The accuracy figures are largely similar to the ones found in the Fabricated Mutation data, and for largely the same reasons. In this case, where the classifier is a simple threshold, it should be obvious that merely shuffling labels cannot

Table 9 Example of a simplified fabricated dataset

Patient label	Number of set bits	Label
Patient0	53	PD
Patient1	57	PD
Patient2	54	PD
Patient3	53	PD
Patient4	60	PD
Patient5	45	Control
Patient6	48	Control
Patient7	49	Control
Patient8	48	Control
Patient9	58	PD
Patient10	63	PD
Patient11	47	Control

Table 10 Example of ECS strategies on a simplified fabricated, shuffled dataset

Patient	Original label	Shuffled label	GA classification	GA threshold
(a)[a]				
Patient5	Control	Control	Control	53
Patient6	Control	Control	Control	51
Patient7	Control	Control	Control	53
Patient8	Control	PD	Control	50
Patient11	Control	Control	Control	52
Patient0	PD	Control	PD	51
Patient1	PD	PD	PD	54
Patient2	PD	PD	PD	52
Patient3	PD	PD	PD	51
Patient4	PD	Control	PD	53
Patient9	PD	PD	PD	51
Patient10	PD	PD	PD	52
(b)[b]				
Patient5	Control	PD	Control	63
Patient6	Control	PD	Control	63
Patient7	Control	Control	PD	47
Patient8	Control	PD	Control	63
Patient11	Control	PD	Control	63
Patient0	PD	PD	Control	62
Patient1	PD	Control	PD	46
Patient2	PD	PD	PD	47
Patient3	PD	Control	PD	47
Patient4	PD	Control	Control	61
Patient9	PD	Control	Control	62
Patient10	PD	PD	PD	47

[a] *Accuracy*: 0.75; in this case, only three patients changed labels. Patients 8 and 0 are close to the original border (they have 48 and 53 set bits), but patient 4 has 60 set bits. It was not enough to change the boundary.
[b] *Accuracy*: 0.33; here we have an unusual case, due to the numerous changes, and the thresholds were always close to the lower or higher ends of the spectrum, meaning that the system was getting stuck in local maxima that allowed it to classify most training instances correctly, yet the out of sample testing had no clear pattern to be exploited.

cause accuracy to fall to random levels—shuffling does not destroy the GA's ability to learn something from the data.

In the original data, any threshold between 50 and 53, inclusive, would work. In the shuffled data, it was often the case that similar thresholds would lead to performance significantly better than random, as can be seen in the run shown in Table 10(a and b).

4.5. Validity of the ECS methodology

The overall result of these various experiments is a clear conclusion that, in spite of the small size of the dataset, the results found by the ECS methodology are meaningful and valid. The cross-validation results are highly statistically significant, and our computational experiments with shuffled and otherwise randomized data provide further evidence that the classification rules found are not artifacts of an overenthusiastic pattern-finding methodology, but rather represent real regularities in the data. It is obvious that greater certainty will be achieved by testing these rules on additional data, and that rules learned on larger datasets will be more confident all around. Even on the current small dataset, however, our methodology would appear to have succeeded in identifying predictive patterns that are significant, non-obvious, and sufficient to allow good classification of prospective samples.

5. Discussion

We describe the application of a GA-based classification scheme to the diagnosis of Parkinson's disease from patterns of mtDNA mutations. Our algorithm achieves 100% accuracy in classifying PD and control samples in both simple and full implementations.

5.1. GA-based algorithms in analysis of complex genetic data

The GA provides not only a means of classifying patients based on subtle patterns in complex

genetic data but also points to regions within genes which are likely to be causally involved in the conditions under examination. Gene segments most frequently used in classification (as plotted in Figs. 1–3) may be more likely to be indispensable for normal functioning of the proteins being examined, and their analysis may be a starting point for further hypothesis-driven research. By concentrating further sequencing efforts on the areas highlighted by the GA, resources could be stretched to allow analysis of larger numbers of patients at a lower cost. Although the height of peaks on the graphs may be related to the level of importance of the given segment for causation of disease, the exact relationship of these quantities is at present unclear. The peaks are influenced by a small number of mutations and therefore prone to being obscured by random features in small samples. While on average the segments within peaks are more likely to be involved in the condition under study than other segments, no single peak may be in pronounced ''definitely involved in PD'', until more patient data are collected.

Further improvements may be made to the algorithm. At present, it provides a binary classification of patients but a version producing a continuous score for each patient, quantifying the mutational burden in regions of interest is possible. A cutoff value resulting in the best separation of patient groups would be then chosen. An advantage of a continuous score lies in the improved ability to detect and analyze borderline cases, which are certain to exist in mitochondrial conditions. The value of weighting by conservation and amino acid dissimilarity could not be convincingly demonstrated in our study, and will need to be further explored.

Some similarity of the current dataset to SNP data may be noted. SNPs are single nucleotide polymorphisms, a form of mutation that is important in research on disorders with complex patterns of inheritance. In both cases, there are mutation patterns that can be assembled in a linear fashion, and the likely influence of single mutations is small. Modifications to the algorithm would have to be made, to allow the use of data from discordant family members, linkage/recombination, and in most cases no weighting of individual mutations would be applied, since most SNPs fall outside coding regions, but a GA-based approach in the spirit of the present work might be of use in this area of research.

5.2. Implications for PD and other mitochondrial conditions

Our results have implications for both PD and for other disorders where heteroplasmic mtDNA mutations may be important, such as sporadic Alzheimer's disease [22,23]. In PD, the finding of specific, narrow regions within complex I genes which have a greater mutational burden in PD than in age-matched controls provides some degree of corroboration to the hypothesis of mitochondrial causation of this disease. The available data are still insufficient to provide a comprehensive picture of the PC-correlated areas, and further sequencing of ND5 and other complex I genes is under way.

Our results indicate that the ECS may be a useful tool for the detection of subtle, non-obvious mutation patterns in mitochondrial microheteroplasmy, which correlate with biochemical and pathological features of samples, and may guide further research into the mechanism of mitochondrial disease.

Note Added in Proof

We conducted an independent, prospective study of another 8 PD and 8 control brains in which we evaluated only the small region of ND5 identified in this study. The presence or absence of amino acid changing mutations in this single region correctly identified 15 of 16 samples. A single control was incorrectly identified as PD [24].

Acknowledgements

This work was supported by the Dana Foundation and by a Morris K. Udall Center of Excellence grant from NINDS (NS39788).

References

[1] Kirby DM, Boneh A, Chow CW, Ohtake A, Ryan MT, Thyagarajan D, et al. Low mutant load of mitochondrial DNA G13513A mutation can cause Leigh's disease. Ann Neurol 2003;54:473–8.
[2] Rossignol R, Faustin B, Rocher C, Malgat M, Mazat JP, Letellier T. Mitochondrial threshold effects. Biochem J 2003;370:751–62.
[3] Simon DK, Lin MT, Ahn CH, Liu GJ, Gibson GE, Beal MF, et al. Low mutational burden of individual acquired mitochondrial DNA mutations in brain. Genomics 2001;73:113–6.
[4] Simon DK, Lin MT, Zheng L, Liu GJ, Ahn CH, Kim LM, et al. Somatic mitochondrial DNA mutations in cortex and substantia nigra in aging and Parkinson's disease. Neurobiol Aging 2004;25:71–81.
[5] Smigrodzki R, Parks JK, Parker WD. High frequency of mitochondrial complex I mutations in Parkinson's disease and aging. Neurobiol Aging 2004;25:1273–81.
[6] Goldberg D. Genetic algorithms in search, optimization, and machine learning. Boston: Addison-Wesley, 1989.

[7] Holland JH. Adaptation in natural and artificial systems. Boston: Bradford Books, 1975.
[8] Parker Jr WD, Boyson SJ, Parks JK. Abnormalities of the electron transport chain in idiopathic Parkinson's disease. Ann Neurol 1989;26:719—23.
[9] Swerdlow RH, Parks JK, Miller SW, Tuttle JB, Trimmer PA, Sheehan JP, et al. Origin and functional consequences of the complex I defect in Parkinson's disease. Ann Neurol 1996;40:663—71.
[10] Sheehan JP, Swerdlow RH, Miller SW, Davis RE, Parks JK, Parker WD, et al. Calcium homeostasis and reactive oxygen species production in cells transformed by mitochondria from individuals with sporadic Alzheimer's disease. J Neurosci 1997;17:4612—22.
[11] Gu M, Cooper JM, Taanman JW, Schapira AH. Mitochondrial DNA transmission of the mitochondrial defect in Parkinson's disease. Ann Neurol 1998;44:177—86.
[12] IUPAC-IUB Commission on Biochemical Nomenclature. A one-letter notation for amino acid sequences. Tentative rules, J Biol Chem 243 (1968) 3557—59.
[13] Andrews RM, Kubacka I, Chinnery PF, Lightowlers RN, Turnbull DM, Howell N. Reanalysis and revision of the Cambridge reference sequence for human mitochondrial DNA. Nat Genet 1999;23:147.
[14] Ng PC, Henikoff S. SIFT: predicting amino acid changes that affect protein function. Nucleic Acids Res 2003;31:3812—4.
[15] Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W, et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res 1997;25:3389—402.
[16] Boeckmann B, Bairoch A, Apweiler R, Blatter MC, Estreicher A, Gasteiger E, et al. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. Nucleic Acids Res 2003;31:365—70.
[17] Xia X, Xie Z. Protein structure, neighbor effect, and a new index of amino acid dissimilarities. Mol Biol Evol 2002;19:58—67.
[18] Hjorth JSU. Computer intensive statistical methods validation, model selection and Bootstrap. Chapman and Hall; 1994.
[19] Goutte C. Note on free lunches and cross-validation. Neural Comput 1997;9:1211—5.
[20] Shao J. Linear model selection by cross-validation. J Am Stat Assoc 1993;88:486—8.
[21] Rissanen J. Stochastic complexity and modeling. Ann Stat 1986;14:1080—100.
[22] Parker Jr WD, Filley CM, Parks JK. Cytochrome oxidase deficiency in Alzheimer's disease. Neurology 1990;40: 1302—3.
[23] Swerdlow RH, Parks JK, Cassarino DS, Maguire DJ, Maguire RS, Bennett Jr JP, et al. Cybrids in Alzheimer's disease: a cellular model of the disease? Neurology 1997;49: 918—25.
[24] Parker WD, Parks JK. Mitochondrial ND5 mutations in idiopathic Parkinson's disease. Biochem Biophys Res Commun 2005;326:667—9.