Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A fuzzy-driven genetic algorithm for sequence segmentation applied to genomic sequences

Elizabeth Jacob [a,*], K.N. Ramachandran Nair [b], Roschen Sasikumar [a]

[a] Computational Modeling and Simulation Unit, National Institute for Interdisciplinary Science and Technology (CSIR), Trivandrum 695019, India
[b] School of Computer Sciences, Mahatma Gandhi University, Kottayam 686560, India

ARTICLE INFO

Article history:
Received 27 June 2007
Received in revised form 26 June 2008
Accepted 27 July 2008
Available online 14 August 2008

Keywords:
Sequence segmentation
Genetic algorithm
Fuzzy fitness finder
Genome
Genes

ABSTRACT

The fuzzy-driven genetic algorithm for sequence segmentation consists of a genetic algorithm whose objective function is driven by a fuzzy fitness finder. The genetic algorithm starts with an initial population of alternate solutions where each solution is a different partitioning of the sequence into segments. The algorithm uses adaptations of the standard genetic operators to reallocate the partitions so as to achieve optimal segmentations. A fuzzy fitness finding mechanism evaluates the fitness values of the evolving segmentations, taking into consideration the combined effect of multiple heterogeneous features that have been identified as governing factors for the formation of the segments. The relationships between segment elements can also be modeled by this novel approach of applying soft computing paradigms to the segmentation of multi-dimensional sequences. The algorithm developed in this work has been successfully implemented for gene sequence segmentation to predict groups of functionally related genes that lie adjacent on the genome sequences of bacterial genomes.

1. Introduction

Sequential datasets are generated from diverse applications like stock market analysis, telecommunications, bioinformatics, text processing, click stream mining and many more. Mining of such data needs the development of techniques that can handle sequences with large number of data points. Sequence segmentation is defined as the division of the sequence into k non-overlapping contiguous segments covering all the data points such that each segment is as homogeneous as possible. Segmentations give a high-level view of the sequence's structure. An ordered set of N numbers $\{a_i: i = 0 \ldots N-1\}$ can be partitioned into k contiguous groups in $\binom{N-1}{k-1}$ ways. An expression for the total number of partitions is calculated as 2^{N-1} where k can take values from 1 to N. For a set of 10 numbers, the number of groups will be 512 but this number increases exponentially to 5×10^{29} approximately, for a data set of 100 numbers. Hence exhaustive enumeration of all possible partitions is not computationally feasible.

The segmentation problem can be solved optimally using dynamic programming which uses recursion. The problem is divided into subproblems which are solved separately and combined to form the original solution. The optimal k-segmentation of $T[1,N]$ is the optimal segmentation of $T[1,j]$ into $k-1$ segments and a single segment that spans the subsequence $T[j+1,N]$. The quadratic running time, however, makes its use prohibitive for long sequences. In the Divide & Segment algorithm (DNS) [1], the main sequence is partitioned into disjoint subsequences and each subsequence is segmented by dynamic programming into k segments. The representatives of the subsequences are concatenated to form a weighted sequence and dynamic programming is again applied on the sequence.

The top-down approach starts with an unsegmented sequence and introduces one boundary at a time [2,3] by splitting the sequence. The bottom-up method begins with all points in different segments. At each step the algorithm merges segments until a k-segmentation is obtained [4,5]. The split and merge is done with the goal of minimization of overall error.

Local search techniques for segmentation [6] use arbitrary or fixed size (oblivious) segments and then keep moving segment boundaries until there is no improvement in the quality of the segments. The sliding window (SW) algorithm scans the sequence from left to right. The left boundary of a segment is fixed and the right boundary is stretched as far as possible. When the error of the segment exceeds a given threshold value then the boundary is fixed and the process repeated till the end of the sequence [3,7,8]. Given a multidimensional time series, the basis segmentation method [9] takes a small set of latent variables (basis set) and use dimensionality reduction techniques like principal component

* Corresponding author. Tel.: +91 471 2515381; fax: +91 471 2491712.
E-mail address: liz.csir@gmail.com (E. Jacob).

analysis to segment the series such that the representatives can be expressed as a linear combination of the latent variables.

Clustered segmentation is for segmentation of multi-dimensional sequences [10], the dimensions of the sequence are grouped to form clusters and each cluster is segmented independently. This produces as many partitions of the same sequence as the number of clusters. Aggregation methods try to make an aggregate partition that agrees as much as possible with the various segmentations of a sequence [11].

The Fuzzy-driven Genetic Algorithm (FGA) proposed in this work belongs to the class of approximate algorithms and generates a set of optimal segmentations. This algorithm has characteristics derived from the different methods for segmentation and hence cannot be classified as belonging to any one of them explicitly. Global search is done by the crossover operator by combining segments from across the genome whereas mutation fixes boundaries of segments by local search. In contrast to clustered segmentation where the dimensions are separated into clusters, the aggregate effect of all the dimensions of the multi-dimensional sequence is found with the help of a fuzzy system.

Sequence segmentation has been successfully applied to problems in biology to predict haplotype block structure, DNA and genome segmentation and regulatory element detection using a probabilistic segmentation model [12–16]. The FGA predicts functionally related gene clusters across a genome. Apart from its application as a tool in computational biology, the technique can be adopted to suit other application domains.

The organization of the paper is as follows. After an introduction to sequence segmentation, Section 2 defines the theory of segmentation in the perspective of the algorithm developed here. Section 3 presents the algorithm in detail. Section 4 is the implementation of the algorithm to genomic sequence segmentation. In Section 5 the analysis of the results of applying the algorithm to three bacterial genomes is discussed. Section 6 gives an analysis of the algorithm. Concluding remarks are given in Section 7.

2. Theory of sequence segmentation

Consider a real multi-dimensional sequence $T = \{t_1, \ldots, t_N\}$ of finite length n consisting of d-dimensional points $t_i \in \Re^d$. Let T_n stand for all such sequences of length N. A k-segmentation S of T can be defined using $(k+1)$ boundaries where the boundary elements belong to T. An error function is a means for assessing the quality of the segmentation of a sequence. An optimal k-segmentation of T can be defined as

$$S_{\text{opt}}(T,k) = \min_{S \in S_{n,k}} E(T,S)$$

where S_{opt} is the k-segmentation S that minimizes the error function $E(T,S)$.

The error of a segment can be measured as the squared Euclidean distance of each point from the representative point for that segment. The error function is decomposable that is it can be decomposed to the error of each segment.

The FGA replaces the error function by a fitness function and instead of minimizing the error tries to maximize the fitness of each segment. The fitness function is also decomposable as the fitness of the segmentation of $T = \sum_{i=1}^{k} \text{fitness}_i$ where k = number of non-unit segments in T. To find the fitness of a segment, the d variables of the d-dimensional sequence are defined as fuzzy variables. Rules from a fuzzy rulebase then combine various inputs of these variables into an output fitness value in percentage for each segment.

3. The soft computing approach to sequence segmentation

The genetic algorithm (GA) [17] has at its heart, a fuzzy logic based fitness evaluator to find the fitness of a set of data objects as a candidate segment, based on multiple criteria or features. The criteria may involve very different representations from numerical values to textual information and their relative importance as well as our level of confidence in them may also vary. Fuzzy logic offers a straightforward method for comparing apples and oranges to arrive at a combined score for a putative segment [18].

3.1. The fuzzy-driven genetic algorithm

The main computational elements of the algorithm are

(1) The Genetic Algorithm (GA) and
(2) The Fuzzy Fitness Finder (FFF).

Fig. 1 gives a flowchart of the FGA.

```
Algorithm 1: The FGA Algorithm

Inputs: a d-dimensional sequence T of size N.
Output: segmentations of T with high fitness values.

(1) Generate initial population of n segmentations for the GA by
    segmenting the sequence
   based on n different cut-off values of a criterion or
   by oblivious segmentations generally of equal length.

(2) Calculate fitness of each segmentation of T
   calculate score of each segment using FFF
   fitness of a segmented sequence = Sum of scores of all its
   segments.

(3) Selection of next generation.

(4) Crossover
   choose a pair of segmentations $T_i$ and $T_j$
   align them and cut at a random position
   recombine the fragments
   accept the two offspring if they are fitter than their parents.

(5) Mutation
   randomly select a segment in a segmentation $T_i$
   rework the segment boundaries if it leads to a fitter
   segmentation.

(6) If terminating condition is true then Exit else go to Step (2).
```

3.2. Selection, crossover and mutation

A standard selection method like the roulette wheel method is used to select individuals for the next generation. The wheel is biased so that segmentations with higher fitness scores get more copies of themselves into the new population.

Crossover is a global search operator that brings in new solutions from the search space by recombination of segments. It is performed by randomly choosing a pair of segmentations from the new population, cutting at a random position on the sequence and combining the fragments (Fig. 2). This involves renumbering of

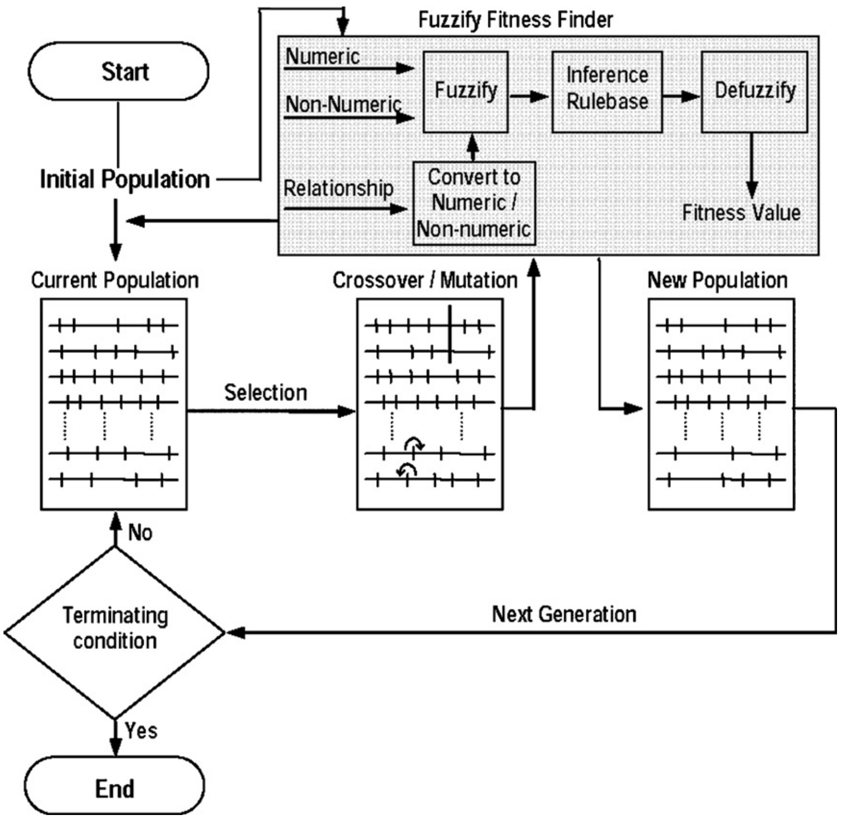


Fig. 1. Flowchart for FGA.

segments in the new individuals. In the example shown, the parents have 11 and 8 segments, respectively. After crossover, the offspring have 9 and 10 segments. If, at least one of the two offspring is better than the parents in terms of fitness, the crossover is said to be successful.

Mutation in FGA is a local search method for improving the fitness of segment members. The mutation operator works at segment boundaries readjusting the breakpoints. Segment borders are randomly selected and are reallocated depending upon how strongly the border data objects are connected to the segment. This leads to switching of segment membership of border objects, merging of segments, delinking to become a unit segment or leaving the segments unchanged (Fig. 3). In the illustration, the mutation site falls at g4. If the fitness of segment (g3,g4) is low and the fitness of segment (g4,g5) is large enough, then g4 switches over to the right segment.

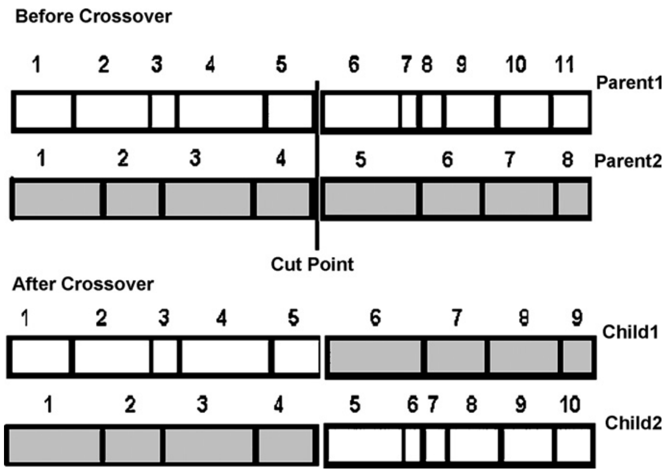


Fig. 2. Illustration of crossover.

The process of evaluation, selection, crossover and mutation is carried on until there is no significant difference in fitness values in successive generations or an upper limit for generations is reached. The last generation consists of the best segmentations found by the algorithm.

3.3. *The fuzzy fitness finder*

3.3.1. *Why fuzzy?*

In a multi-dimensional sequence, a segmentation may be best *w.r.t.* one variable but not so *w.r.t.* the other variables. In the classical method of objective weighting, each variable is given a weight and the weighted sum of the fitness of individual variables is taken as the overall objective function. The weights are a measure of the significance of a variable in comparison to the other variables. For example if one weight is two times the other, it implies that the first variable is doubly significant compared to the second. However, a set of fixed weights leads to a constant interrelationship. If values for one or more of these criteria are not available, then the total fitness score will drop drastically. Actually, some of the other criteria may be enough to give a high combined score. Under different combinations of inputs, the weight values can vary.

In a fuzzy-logic based system, a weighting function is replaced by a set of rules. A variable can take a range of values in an interval. The variable, if modeled as a fuzzy variable can belong to different classes in the interval with different probabilities. Different combinations of different variables (fuzzy inputs) join together to produce unique effects. A fuzzy logic inferencing system can

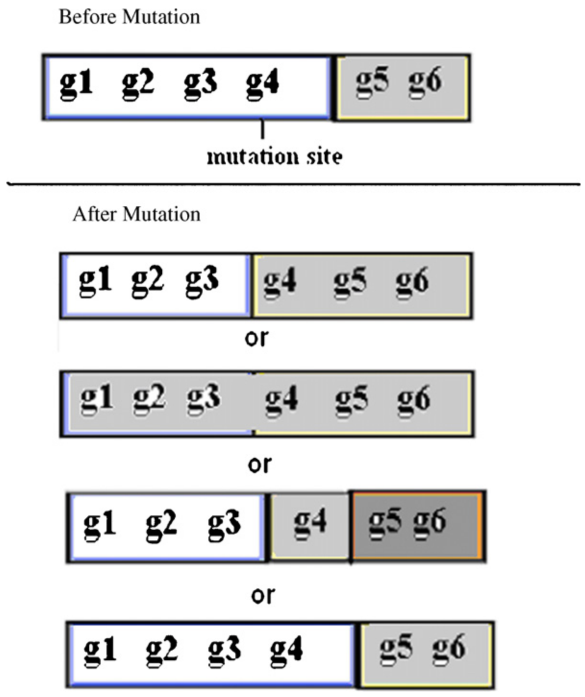


Fig. 3. Illustration of mutation.

capture the complex interactions among the variables which is not possible by assignment of fixed weights to the inputs.

```
Algorithm 2: Fuzzy Fitness Finder

Inputs: a segmentation of sequence T
Output: fitness score of the segmentation of T

(1) For each of the d variables, calculate a crisp value for
    every segment
  numerical values: average of values of segment members
  non-numerical values: convert to categories or numerical
  value
  relationships between segment elements: convert to
  numerical value by (sum of pairwise scores of segment
  members)/number of pairs.

(2) Design of fuzzy membership functions for each of the
    d variables.

(3) Fuzzification of the values of the d variables of every segment.

(4) Fuzzy inference mechanism
  Construction of fuzzy rulebase
  IF variable1 is very high AND ... variable d is low
  THEN fitness score is high
  rules fire when antecedents match.

(5) Defuzzification of fitness scores
  Consequents of the fired rules defuzzified to yield a crisp fitness
  score (in %) for a given segment.

(6) Fitness score of a segmentation of T = summation of fitness
    of non-unit segments.
```

The FFF is applied to every segment of all individuals of the initial population. In each generation, variation is brought about by crossover and mutation. In crossover, only if the cut-point falls inside a segment, the fitness values have to be recalculated. Similarly, fitness values have to be re-calculated for segments affected by mutation only. Hence, the FFF is called only if genetic operators disturb the segments.

4. Application to gene sequence segmentation

4.1. The system

Different segments of the DNA sequence, code for different proteins and these segments are called genes. The code gets transcribed and translated into proteins when the genes are "expressed". The full sequence containing all the genes and intergenic regions (stretches of bases that have not been identified as genes) is called the genome of an organism.

It has been observed both experimentally and computationally that in lower organisms like bacteria and to some extent in higher organisms, genes that code for related functionality huddle together on the genome. Evolution favours such grouping as it is easier for these genes to get expressed (decoded) together when they are called upon to carry out the particular function. Groups of adjacent genes that are expressed together are called "operons".

The genome can be viewed as a sequence of genes g_1, g_2, g_3,... g_n with each gene separated from the next by an intergenic gap of bases that varies from gene to gene. In order to group genes into operons, some of the factors that could be responsible for keeping the genes together as operons are commonality of metabolic pathways and similarity of protein function. The segments from the FGA are candidate operons. As experimental detection of operons is time consuming and difficult to implement in the laboratory, computational algorithms for operon prediction are being recognized as a tool to discovery of important gene groups in newly sequenced microbial genomes.

4.2. Implementation

Genomes of three bacteria were studied using the FGA algorithm

I. Scoring criteria taken for grouping genes into operons are
   (a) intergenic distance (number of bases in between two genes),
   (b) participation in the same metabolic pathway
   (c) conservation of the proximity of a gene pair across many genomes
   (d) similarity of protein function.
II. The initial population is created by partitioning the genome using 10 different threshold intergenic distance values from 0 to 600 bases. For a threshold of 200 bases, if the gap between two genes is greater than 200 bases, then a new segment is initiated.
III. Calculation of crisp input values for the criteria.
   (a) The score based on intergenic distance is the average intergenic distance between consecutive genes within an operon.
   (b) Commonality of metabolic pathway / protein function and conservation If two genes in an segment have a common pathway or protein function, the score for that pair is taken as 1 or else as 0. If there are *m* genes in a segment, all pairs of

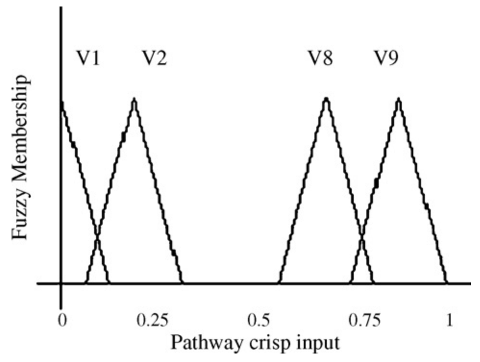


Fig. 4. Fuzzy membership functions for pathway.

genes are scored as above and divided by ${}^{m}C_2$ to give a crisp value between 0 and 1. Thus if all pairs have a common pathway, the pathway score for the segment will be 1. For conservation, pair-wise scores are available in literature.

IV. Fuzzification of inputs

The fuzzy membership functions for each variable is defined by triangular sets (V1, V2,...) where V1, V2 have linguistic interpretations like extremely low, very low, high, etc.

Fig. 4 shows the fuzzy membership functions for the fuzzy variable, metabolic pathway. Here, scores close to one implies that the two genes have a good possibility of being together in the same segment, i.e. they participate in similar metabolic pathways.

V. Fuzzy inference

The inference engine works from a rulebase. The output value which is the fitness of a segment is in percentage. The output range (0–100) is divided uniformly into fuzzy sets (V1, V2,...). The rules are of the form:

IF pathway is V8 AND conservation is V8 THEN fitness is V7.

For the three criteria case onwards, due to the large number of rules, the inference engine is cascaded or it combines the scores two at a time. It first combines the distance score with pathway score. The combined score of distance and pathway criteria is then combined with the conservation score to get the final fitness.

VI. Defuzzification of output

The final fitness scores are de-fuzzified using the root-sum-squares method to yield a crisp fitness value.

4.3. Program parameters

The parameters that the designer can select are the parameters of the GA and the FFF. Parameters chosen are:
Population size = 10.

As crossover is not taken to be a random phenomenon, there is no need to define a probability for crossover which is taken as a sure event with probability equal to one.

Probability of mutation = 0.1.

For mutation, the threshold values are (15, 75). Therefore, the pair-wise fitness is considered significant if it is more than 75 and insignificant if less than 15.

The maximum number of iterations = 50.

The number of fuzzy sets for inputs as well as output = 9.

Fuzzy membership functions for the variables and the rules are designed by intuition.

4.4. Aggregation of segmentations

A 'best of *n*' genome map is constructed by gene-by-gene analysis of segment fitness. Each gene is taken and it is seen in which of the *n* genome maps, it has the best fitness. The genes of that segment are chosen. For example in *Escherichia coli*, gene b0001 may have maximum fitness when grouped as b0001 and b0002. b0002 could have maximum fitness when it is grouped as b0002, b0003 and b0004. So, b0002 appears in two segments and the biologist can decide which of these to choose. The *n* genome maps generated by the algorithm have no overlaps but the post-processing for aggregation causes overlaps.

5. Analysis of results

Genome data are taken from public domain databases [19,20].

5.1. Case 1: E. coli

E. coli K12 consisting of about 4400 genes are friendly bacteria that inhabit the intestine but sometimes turn virulent. The genes are named as b#### (e.g. b0001).

At first only one criterion (intergenic distance) was used. Then the second criterion of conservation across genomes was added and operons generated. For the three criteria case, commonality of metabolic pathways was added. In the four criteria case, protein similarity criterion was added.

Most of the experimentally predicted operons were discovered exactly by our algorithm. There are currently 237 known operons in *E. coli* of which our method predicted 213 operons (sensitivity = 90%). The *mhp* operon (b0347, b0348, b0349, b0350, b0351 and b0352) and the *Men* operon (b2260, b2261, b2262, b2263, b2264 and b2265) were predicted with 100% fitness value by the FGA. In certain cases, genes that are not predicted by experiment appear inside operons. This could lead to the possible annotation of genes whose functions are currently unknown. The biological results using the method have been published [21]. A few predicted operons alongwith their fitness values are shown in Table 1.

The operon b0001...b0004 is an experimentally predicted operon of four genes but our algorithm predicts only three of them to be in one operon with a fitness value of 100%. The intergenic distance by definition is the number of base pairs between genes and their average is a numeric value. As the four genes have a common pathway (eco00260 for metabolism) and protein (EC1.5.1.8 oxidoreductase), their scores are one. The conservation score available between pairs of genes is averaged to get 0.82. The

Table 1
Operons predicted for *E. coli*

Experimentally predicted operons	Predicted by FGA	Dist	Path	Cons	Protein	Fitness (in %)
b0001to b0004 (4 genes)	b0002 to b0004 (3 genes)	1.5	1.0	0.82	1.0	100
b0081 to b0095 (15 genes)	b0081 to b0096 (16 genes)	13.4	0.17	0.07	0.52	55
b2019 to b2026 (8 genes)	b2019 to b2026 (8 genes)	14.88	0.78	0.11	1.0	100
b3435 to b3437 (3 genes)	b3435 to b3437 (3 genes)	6.0	0.0	0.76	1.0	65

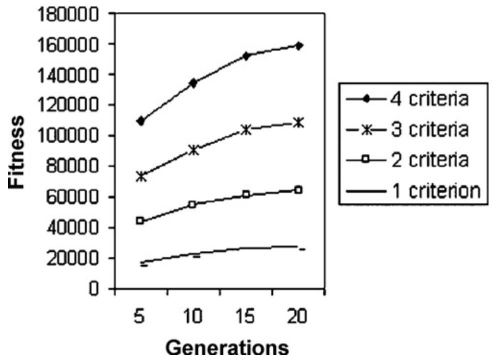


Fig. 5. Fitness vs. number of variables in *E. coli*.

relationships between the genes are converted to numerical values representing the segment. The FFF calculates the fitness score using the rules that combine these criteria. Fig. 5 is a graph showing the relationship between the number of criteria and fitness values for *E. coli* taken at four intermediate points (after 5, 10, 15 and 20 generations) of a run.

The four criteria are added in the order intergenic distance, pathway, conservation and protein function. It can be noted that additional features improve the fitness value. To start with, the initial fitness is higher when a new criterion is added. The reason for this is that when only one feature is considered, the presence or absence of one feature results in small segments with high fitness values (100%), however, many segments are unit segments which do not have a fitness contribution. When there are more features, there are more multi-gene clusters formed because the contribution of many features brings together more genes into clusters. There will be more clusters with less than 100% fitness and the summation of these leads to a higher total fitness value. Though, the rule seems to imply that more criteria means better fitness values, this cannot be generalized.

5.2. Case 2: Bacillus subtilis

B. subtilis is a bacteria that promotes plant growth and has been shown to increase crop yield. They live at the interface between the plant roots and the surrounding soil. The sequencing of its genome was completed in 1997. The genome has 4200 genes. Genes of *B. subtilis* are named as BG#####.

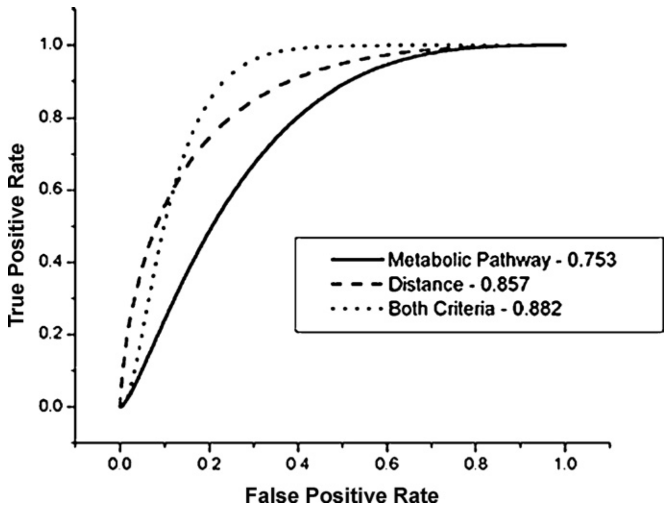
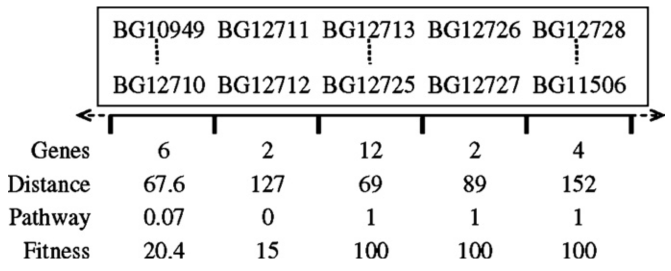


Fig. 6. ROC curves for *B. subtilis*.



Genes	6	2	12	2	4
Distance	67.6	127	69	89	152
Pathway	0.07	0	1	1	1
Fitness	20.4	15	100	100	100

Fig. 7. Part of the operon map of *B. subtilis*.

Data for *B. subtilis* was available only for intergenic distance and metabolic pathway. An ROC (Receiver Operating Characteristic) curve has been used to evaluate the overall accuracy and predictive value of the method. The ROC curve is obtained by plotting the True Positive rate (TP) against the False Positive rate (FP) where TP = Fraction of experimental linkages that are predicted by the method and FP = 1 − fraction of experimental borders predicted by the method.

The area under the ROC curve is a measure of the predictive value of the method. The ROC curves for *B. subtilis* using the distance and metabolic pathway criteria individually and taken together are shown in Fig. 6. The metabolic pathway data being scantier and probably less reliable for *B. subtilis* compared to *E. coli*, this criterion makes only a small improvement to predictions based on distance alone. Using both criteria, the area under the ROC curve increases to 88.2%. The same rules that were used for *E. coli* were used for scoring. Experimental data received in personal communication from Shujiro Okuda of Kyoto University (2004) was used. Fig. 7 shows a stretch of gene segments on the genome with their scores for pathway and distance before fuzzification and fitness values.

5.3. Case 3: Mycobacterium tuberculosis

M. tuberculosis bacteria causes tuberculosis. Thought to have evolved from a soil bacterium, it evolved to infect cows and then made the jump to humans when animals were domesticated 10,000 years ago. Of late, it has been found to have a deadly partnership with the AIDS virus. There are 3924 genes of *M. tuberculosis* named as Rv####.

A computational tool proves to be very useful for predicting operons in organisms, for which no training set of experimental operons is available. *M. tuberculosis* does not have experimental operons but Zheng et al. [22] have made predictions based on metabolic pathways alone using a graph-theoretical approach. Our predictions were compared with these and it was found that 80 of the 89 operons were predicted by our method also.

Some of the predictions comparing operons predicted by the two methods are listed in Table 2. While some operons are predicted exactly by both methods, some others are predicted with more or less genes and some are not predicted at all by Zheng et al.

6. Algorithm analysis

6.1. Segment quality index

In order to evaluate the quality of segments generated by the algorithm, there are standard cluster validity indices. The Davies–Bouldin (DB) index [23] is a function of the ratio of the sum of "within-cluster" scatter to "between-cluster" separation. The DB analysis was suitably modified to determine the equivalent of

Table 2
Operons predicted for *M. tuberculosis*

Predicted operons, Zheng et al.	Predicted by FGA	Fitness (in %)
Rv0407 to Rv0408 (2genes)	Rv0407 to Rv0409 (3genes)	55
Rv3145 to Rv3158 (14genes)	Rv3145 to Rv3151 (7 genes)	100
Rv2064 to Rv2066 (3genes)	Rv2064 to Rv2066 (3genes)	100
Not predicted	Rv0254 to Rv0255 (2 genes)	100
Not predicted	Rv0880 to Rv0882 (3 genes)	67

"within-cluster" scatter and "between-cluster" separation.

$$\text{Segment Quality Index (SQI)} = \frac{1}{n_c}\sum_{i=1}^{n_c} R_i \tag{1}$$

where n_c = number of segments

$$R_i = \max_{\substack{j=i-1\\ j=i+1}} \frac{S_i}{d_{ij}}$$

$$= \max_{\substack{j=i-1\\ j=i+1}} \frac{100-f_i}{100-p_{ij}}$$

where S_i is the scatter within the ith segment, d_{ij} is the separation between the ith segment and the jth segment.

The maximum is taken over the left and right ratio values. The fuzzy fitness score (f_i) for the segment is a measure of the similarity of the genes constituting the segment with respect to the multiple features compared. Thus a high fitness score is equivalent to a low scatter within the segment and vice versa. Therefore $(100 - f_i)$ is considered equivalent to the scatter S in our calculation. Similarly the separation between two segments d_{ij}, is taken as $(100 - p_{ij})$ where p_{ij} is the pair-wise fitness across bordering elements of segments on the right and left. Therefore, more the pair-wise fitness at the border, lesser will be the separation between the two segments.

Fig. 8 plots the SQI values for criteria starting from the criterion of intergenic distance alone and adding conservation, common pathway and similarity of protein function. The DB index has been plotted at the end of 20 generations. Minimization of CQI achieves better segmentation. The value of the index is minimum when all four scoring features are used for segmenting (zero) and maximum when only the first feature, intergenic distance is used (0.2288). In this application, segments improve when more criteria are taken into the study.

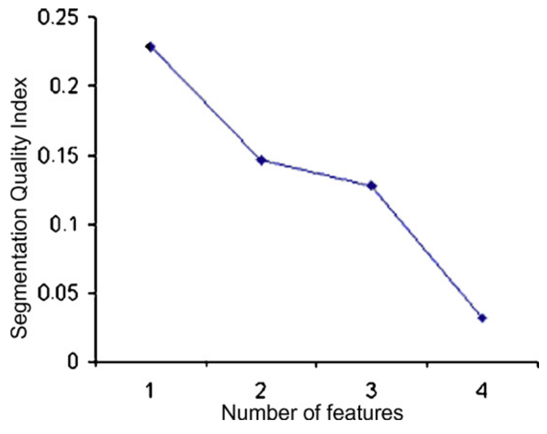


Fig. 8. Variation of segmentation quality index with the number of variables.

6.2. Complexity calculations

6.2.1. Time complexity

An expression for time complexity has been derived as follows.

$$\begin{aligned}\text{Time Complexity} = {} & \text{Complexity(initial population)}\\ & + \text{Complexity(an iteration)}\\ & \times \text{number of iterations}\end{aligned}$$

Let n_{iter} = number of iterations of the algorithm; n = population size or number of different segmentations; N = total number of data items in the sequence; m = average segment size excluding single unit segments; n_f = number of features; k = average number of non-single segments in a segmentation of the sequence; K = average number of segments in an individual including single segments; m_p = average number of segments affected due to mutation in an individual.

$$\begin{aligned}&\text{Complexity(initial population)}\\ &= \text{Complexity(creating segments)}\\ &\quad + \text{Complexity(computing fitness values of the segments)}\\ &= nN + ({}^mC_2 \cdot n_f + n_f - 1)\cdot k \cdot n = O(n\cdot N + {}^mC_2 \cdot n_f \cdot k \cdot n)\end{aligned} \tag{2}$$

If data items in a segment have to be compared pair-wise then ${}^mC_2$ comparisons have to be made to arrive at a crisp value for each feature. For n_f number of features, the fuzzy fitness finder is executed $(n_f - 1)$ times for each segment. $(n_f - 1)$ can be ignored as ${}^mC_2 n_f > (n_f - 1)$ since ${}^mC_2 \geq 1$ (as $m \geq 2$).

$$\begin{aligned}\text{Complexity(an iteration)} = {} & \text{Complexity(selection}\\ & + \text{crossover} + \text{mutation)}n + {}^mC_2\\ & \cdot n_f \cdot n + {}^mC_2 \cdot n_f \cdot m_p \cdot n O({}^mC_2 \cdot n_f\\ & \cdot m_p \cdot n)\end{aligned} \tag{3}$$

From (2) and (3), an estimate of the time complexity is given by $O(n{\cdot}N + {}^mC_2{\cdot}n_f{\cdot}n(k + n_{\text{iter}}{\cdot}m_p))$.

For the first generation, all segments in each individual are evaluated for their fitness. Hence the main computational burden is incurred at the start. In following iterations, only the segments that are affected by crossover and mutation have to be recalculated.

6.2.2. Space complexity

Space Complexity is calculated as: Space Complexity = Space for data items + Space for segment information.

Every data item has criteria information. For every individual, each data item will have a segment number.

$$\text{Space for data items} = n_f \cdot N + n \cdot N \tag{4}$$

Pertaining to each segment, there are n_f scores for criteria values and one value for segment fitness. For storing the values of the whole population, the space needed has to be summed over all segments of an individual and then over all the individuals.

At every instant, the values of the new population are being calculated from the old population. Hence the same space is needed for the new population also.

$$\text{Space for segment information} = 2K \cdot n \cdot (n_f + 1) \tag{5}$$

Adding (4) and (5), an estimate of space complexity is given by $S_p = N(n_f + n) + 2 \cdot K \cdot n(n_f + 1)$.

Genetic algorithms are sensitive to control parameters like population size, probabilities of crossover, mutation, etc. Tuning of these parameters is required for each problem.

6.3. Comparison with other algorithms

Computational prediction of operons is based on one or more of the features that operons are known to have. The notable efforts in this direction have been:

(1) Estimation of the likelihood that genes appear together across genomes [24,25].
(2) It is observed that genes within operons have much shorter intergenic distance in base pairs than genes at the borders of operons. Frequency distribution of distance between genes shows that within operons, peaks are close to zero whereas there are higher peaks at the boundaries [26]. The method identifies about 75% of known operons in the *E. coli* genome.
(3) Probabilistic approach to predicting operons using Bayesian networks [27]. The learning method uses features like length and spacing of genes, codon usage, transcription signal features and gene expression features from microarray experiments to train the Bayes net from which a new candidate operon can be classified as operon or non-operon. They were able to identify over 78% of *E. coli* operons at a 10% false positive rate.
(4) Graph-theoretic methods [22,28] where genes are represented as nodes and their relationships as edges.
(5) The Hidden Markov Model [29] of *E. coli* gene structure from which operons were predicted.
(6) Bayesian classification scheme from gene expression data on *E. coli* using the correlation between expression ratios of adjacent genes [30].

The FGA adds itself to the repertoire of computational methods for operon prediction. The algorithm scores over the existing computational methods by

(1) providing a whole genome map whereas ordinarily only pairwise comparisons between genes emerge and
(2) modeling the inexact nature of the various factors that contribute to the formation of gene clusters.

FGA uses available data and simple rules to determine operons and does not need any prior training.

As a method for sequence segmentation, the FGA has to be compared with the standard algorithms. The dynamic programming algorithm has a time complexity of $O(N^2k)$. The DNS divide and segment method has subquadratic running time $O(N^{4/3}k^{5/3})$. The top-down approach has a running time of $O(Nk)$ and its complement the bottom-up method runs in $O(N \log N)$ time. From Eqs. (2) and (3), if the population size is kept small, the FGA will have linear complexity in N. If the segment size (m) is reasonably small then ${}^{m}C_2$ combinations within a segment to find the values for input criteria will not be a burden. The term ${}^{m}C_2$ will be replaced by m, if only adjacent members of a segment are compared. For large data mining applications, the data can be split into two or more subsequences which will reduce the space as well as time complexity.

7. Conclusions

Currently, the main contribution of this work is to the field of computational biology in the form a segmentation technique to search for structure in genomic sequences. However, the methodology holds promise for extension to other application areas. The following are the major contributions made by the algorithm to the general sequence segmentation problem.

The scheme of segmentation starts with a set of solutions, the same data set partitioned in different ways and makes a parallel search by mixing the individuals and changing the limits of the partitions. As many solutions are studied, the chances of reaching a sub-optimal solution are less than in the conventional methods.

The FGA algorithm is designed to handle real-world problems that are not necessarily number sequences. Non-numerical data is not easy to handle. In a one-dimensional sequence taking nominal values, a segment can be defined by consecutive points taking the same value but in the multi-dimensional case, standard algorithms cannot consider all dimensions simultaneously. They also cannot handle a mix of nominal and numerical dimensions. Fuzzy logic gives a method to combine criteria of diverse types and find their collective contribution to the formation of a segment.

The soft computing based hybrid approach to sequence segmentation, on the one hand allows the search for many solutions at the same time and on the other, it finds a way to encapsulate the interrelationships of the segment members. Development of fuzzy rules being highly domain-specific, the algorithm can be applied to sequence segmentation problems in other application areas if the domain expert can conceive and develop the fuzzy rules.

Acknowledgements

We acknowledge Mr. Alex Andrews P. for expert help with the figures. The authors thank the reviewers for their constructive comments that have improved the quality and presentation of the paper.

References

[1] E. Terzi, P. Tsaparas, Efficient algorithms for sequence segmentation, in: Proceedings of the SIAM International Conference on Data Mining, 2006.
[2] D.H. Douglas, T.K. Peucker, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, Canadian Cartographer 10 (2) (1973) 112–122.
[3] H. Shatkay, S.B. Zdonik, Approximate queries and representations for large data sequences, in: Proceedings of the International Conference on Data Engineering (ICDE), 1996, pp. 536–545.
[4] E.J. Keogh, P. Smyth, A probabilistic approach to fast pattern matching in time series databases, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 1997, pp. 24–30.
[5] E.J. Keogh, M.J. Pazzani, An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 1998, pp. 239–243.
[6] J. Himberg, K. Korpiaho, H. Mannila, J. Tikanmaki, H. Toivonen, Time series segmentation for context recognition in mobile devices, in: Proceedings of the IEEE International Conference on Data Mining (ICDM), 2001, pp. 203–210.
[7] E.J. Keogh, S. Chu, D. Hart, M.J. Pazzani, An online algorithm for segmenting time series, in: Proceedings of the IEEE International Conference on Data Mining (ICDM), 2001, pp. 289–296.
[8] A. Koski, M. Juhola, M. Meriste, Syntactic recognition of ECG signals by attributed finite automata, Pattern Recognition 28 (12) (1995) 1927–1940.
[9] E. Bingham, A. Gionis, N. Haiminen, H. Hiisila, H. Mannila, E. Terzi, Segmentation and dimensionality reduction, in: Proceedings of the SIAM International Conference on Data Mining (SDM), 2006.
[10] A. Gionis, H. Mannila, E. Terzi, Clustered segmentations, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), Workshop on Mining Temporal and Sequential Data (TDM), 2004.
[11] T. Mielikainen, E. Terzi, P. Tsaparas, Aggregating time partitions, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2006.
[12] M. Koivisto, M. Perola, T. Varilo, et al., An MDL method for finding haplotype blocks and for estimating the strength of haplotype block boundaries, in: Pacific Symposium on Biocomputing, 2003, 502–513.
[13] W. Li, DNA segmentation as a model selection process, in: International Conference on Research in Computational Molecular Biology (RECOMB), 2001, 204–210.
[14] V. Ramensky, V. Makeev, M.A. Roytberg, V. Tumanyan, DNA segmentation through the Bayesian approach, Journal of Computational Biology 7 (1/2) (2000) 215–231.

[15] M. Salmenkivi, J. Kere, H. Mannila, Genome segmentation using piecewise constant intensity models and reversible jump MCMC, in: Proceedings of the European Conference on Computational Biology (ECCB), 2002, pp. 211–218.
[16] H.J. Bussemaker, H. Li, E.D. Siggia, Regulatory element detection using a probabilistic segmentation model, in: Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology, 2000, pp. 67–74.
[17] D.E. Goldberg, Genetic Algorithms in Search Optimization and Machine Learning, Addison-Wesley, New York, 1989.
[18] E. Cox, The Fuzzy Systems Handbook, AP Professional, New York, 1994.
[19] http://www.genome.ad.jp/kegg.
[20] http://www.tigr.org.
[21] E. Jacob, R. Sasikumar, K.N.R. Nair, A fuzzy-guided genetic algorithm for operon prediction, Bioinformatics 21 (2005) 1403.
[22] Y. Zheng, et al., Computational identification of operons in microbial genomes, Genome Research 12 (2002) 1221.
[23] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Transactions on Pattern Analysis and Machine Intelligence 1 (1979) 224–227.
[24] R. Overbeek, M. Fonstein, M. D'Souza, G.D. Pusch, N. Maltsev, The use of gene clusters to infer functional coupling, Proceedings of the National Academy of Sciences, United States of America 96 (1999) 2896–2901.
[25] M.D. Ermolaeva, O. White, S.L. Salzberg, Prediction of operons in microbial genomes, Nucleic Acids Research 29 (5) (2001) 1216–1221.
[26] H. Salgado, G. Moreno-Hagelsieb, T.F. Smith, C.V. Julio, Operons in *E. coli*: genomic analysis and predictions, Proceedings of the National Academy of Sciences, United States of America 97 (12) (2000) 6652–6657.
[27] J. Bockhorst, M. Craven, D. Page, J. Shavlik, J. Glasner, A Bayesian network approach to operon prediction, Bioinformatics 19 (10) (2003) 1227–1235.
[28] A. Nakaya, S. Goto, M. Kanehisa, Extraction of correlated gene clusters by multiple graph comparison, Genome Informatics 12 (2001) 44–53.
[29] T. Yada, M. Nakao, Y. Totoki, K. Nakai, Modeling and predicting transcriptional units of *E. coli* genes using hidden Markov models, Bioinformatics 15 (12) (1999) 987–993.
[30] C. Sabatti, L. Rohlin, M. Oh, J.C. Liao, Co-expression pattern from DNA micro array experiments as a tool for operon prediction, Nucleic Acids Research 30 (13) (2002) 2886–2893.