An Evolutionary Algorithm for General Symbol Segmentation

Stephen Pearce
University Of Guelph
Guelph Ontario Canada
spearce@uoguelph.ca

Maher Ahmed
Assistant Professor, Wilfrid Laurier University
Waterloo Ontario Canada
mahmed@wlu.ca

Abstract

A new system is presented for general symbol segmentation, which is applicable for segmentation of any connected string of symbols, including characters and line diagrams. Using a powerful graph representation and an evolutionary algorithm framework, segmentation hypotheses are initialized and evolved towards a fully segmented and recognized string. The evolutionary segmentation was tested in many domains including connected digits, connected characters and simple circuit diagrams. The performance of the evolutionary algorithm depends heavily on the symbol recognition system used.

1. Introduction

The problem of separating unknown connected symbols takes many forms. The unknown symbols could have a related meaning in any context imaginable. Most commonly, these symbols are adjacent characters in a handwritten string, but could also be components in a circuit diagram or glyphs in a mural.

The problem of recognizing digits, characters, and other symbols has been addressed by many researchers [1]. This system and others that accomplish the same goals, achieve high accuracy but each one relies on the same assumption, that the input symbol is isolated and free of noise.

In reality, most handwriting runs together, contains broken characters, or is slurred by other noise or lines on the paper. The effects of noise can be minimized through image processing techniques, but characters running together presents a paradox. The paradox is that the individual symbols can not be recognized until they are separated. Until they are recognized, however, there is no reliable information that would help to perform the segmentation. Information that would help includes what the symbols are, or even how many are connected.

2. Background and review

Previous segmentation algorithms have attempted to locate symbols based on properties of the domain. These properties usually define the domain and the places where the algorithm would be applicable and may include either the arrangement of the symbols or common features in the symbol set.

Character segmentation involves separating complicated characters arranged horizontally on a baseline. Connected characters are often placed so close that they are touching, overlapping, sharing a line in the image, or leaning over each other. In these cases there are no extra line segments to remove that will separate the characters.

Different character segmentation systems were developed by Strathy and Suen [2] and Parizeau and Plamondon [3] that at some levels are quite similar to each other. In both systems, segmentations are generated by determining an appropriate position to break the connected string.

In [2] the string is surveyed and a predetermined number of potential cuts are generated directly from features in the string. Sub-strings that exist between the cuts are treated as potential symbols. The method in [3] describes the connected string in terms of characteristic points and their surrounding symbol primitives. The primitives are grouped and regrouped until they can be recognized or accepted by some symbol classifier

The system in [2] would encounter difficulties with characters that are doubly connected. In this case, two cuts would have to be made on a single end of the segmentation hypothesis. Similarly, the primitives used in system [3] may not describe all possible symbols or features in a string, since a primitive set designed for the Latin alphabet may not be able

Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)
0-7695-1960-1/03 $17.00 © 2003 IEEE

to describe some symbols in other alphabets such as Chinese or Arabic.

Parizeau et. al. introduce a method in [4] that uses a window of attention to isolate known characters in a string. This method is focused less on dividing the connected string, and more on discovering small portions that are recognizable. The image that appears inside this window is submitted to a character classifier in an attempt to match it to a known pattern. Two artificial neural networks are used in this system. One is a detector network that is used to recognize the characters from a training set, and the other is a locator network that is trained to move and resize the window from a partial view of a character to a full bounding box of the character.

Connected symbols in a flow diagram are generally small, simple symbols arranged both horizontally and vertically and are connected by long non-symbol lines. Flow diagrams appear in many forms including circuit diagrams, software design, project planning, and chemical plant flow diagrams. Segmentation of logic circuits was addressed by Yu et. al. [5] with a method that isolated logic and circuit symbols by looking for small polygons in the line image.

In their method, Yu et. al. first determine the maximal lines which represent continuous strokes that cross or intersect with other lines. Each line is then evaluated based on criteria relating to shape, size, orientation, and end points.

The evaluation determines whether the line is probably a symbol line, or probably a connection line and each line is given a tentative assignment into one of these groups. The probable symbol lines are then grouped in an attempt to match the symbols they represent. The algorithm continues to form new groups, sometimes reassigning a line from being a probable symbol to being a probable connection line or vice versa, until symbols can be matched, and suitable connection lines are found between the symbols.

This approach was appropriate for diagrams with the properties that the symbols contain small loops, and that the connections are long compared to the dimensions of the symbols. This is because of the rule based methods for assigning lines to groups. In any other domain than flow diagrams, the assignment rules would be inappropriate and the system would fail.

3. A Common Approach

3.1 Overview

The goal of the proposed system is to combine some strengths of the previously described segmentation approaches into an evolutionary algorithm framework. This new system will be capable of working in any symbol segmentation domain, whether it is characters or flow diagrams.

In this framework, the connected string input will be represented as an abstract graph. Each individual will represent a partition of the graph. The individuals will be evaluated with the help of an external symbol recognition system that is independent of the segmentation algorithm so that this technique can be used in different problem domains with very little change. The fate of the evolution is to create an individual that has a part (or segment) matched that corresponds to each character in the connected string.

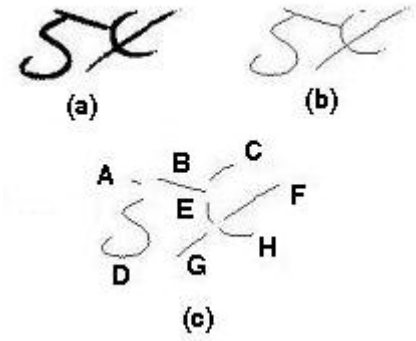

Figure 1. A sample from CEDAR. (a) The original, (b) a thinned version, and (c) the thinned version decomposed into graph form.

Before the evolutionary algorithm may be initialized it is necessary to perform a few pre-processing operations. The connected string must be represented as a binary digital image, figure 1(a), so that the image may be thinned using the algorithm in [6], to produce results similar to figure 1(b). Thinning the image into a line image with a width of one pixel is necessary for the graphing algorithm described in the next section.

3.2. Individual Representation

The first step in representing a potential segmentation is to represent the input image in the form of a graph, figure 1(c). Nodes are placed at each pixel that has more or less than

two neighbours, and edges are traced as an eight-directional chain code from node to node. The edges of the graph are then labled so that a sub-graph may be drawn as it appears in the whole based on an edge set.

An individual consists of a partition of this graph, so that each part represents one symbol out of the entire string. Each part may be redrawn as it appeared in the connected string, so that it represents a clean isolated character that the recognition system assumes as input. The recognition system may either accept or reject the part, and thus an individual is worth more if it has more parts accepted.

3.3. Population Initialization

In theory, the individuals can be initialized randomly. It is common with evolutionary algorithms, however, that if the individuals start near good solutions, the algorithm will perform better. For this reason a new algorithm was designed that would roughly separate the connected string in varying size portions.

The initialization algorithm first found the eastern and western-most degree one nodes. From there, a breadth first search was performed from both nodes to determine the depth of each edge from either end. The maximum depth from either end was recorded as d_{east} and d_{west}.

The only parameter for the algorithm is a balance parameter, in the range zero to one. The parameter p was used to calculate the cut off depth d_{cut} for one segment in the individual.

$$d_{cut} = pd_{east} \qquad (1)$$

$$d_{cut} = (1-p)d_{west} \qquad (2)$$

If the parameter was below one half, then the east segment would form first using equation (1), if it was above one half, the west individual would form first using equation (2). In either case, all edges found with a depth less than or equal to d_{cut} were added directly to the appropriate segment. The other edges are assigned based on their position in the search order, with edges being assigned to the part that they are closer to, and a third made up out of the any edges equally close to either end.

The hypothesis in figure 2(a) consists of segment A made up of three edges, segment B made up of four edges and segment C is the remaining edge.

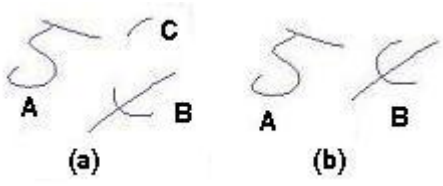

Figure 2. The evolution of figure 1. (a) An initial hypothesis and (b) the final result.

3.4. Population Evolution

In general, evolutionary algorithms evolve the population with the use of crossover and mutation operators controlled by probabilities. In this system, evolution is controlled by opportunity. During each epoch, an attempt is made to find compatible individuals for crossover. Also, it is unlikely that every individual will contain useful material at the outset, or at any time in the evolution, thus it is necessary to allow for invalid individuals that are not close to a solution.

3.4.1 Mutation. The mutation operator described here is a directed one. When it is applied to an individual, a segment is chosen at random that is not already matched. Inside this segment, a degree one node is chosen and the connecting edge moved to another unmatched part of the same individual.

The directed mutation operator described above was found to be insufficient for some cases. So a second level was added to it, to compensate for the more difficult cases and is only applicable to strings where at least one symbol is already matched and there are expected to be non-symbol lines in the string.

The second level directed mutation consists of a mutation that pushes the individual towards a state where it's segments are equal in area. A first step in the second level, is to determine the proper symbol area from the previously matched segment. From this point, one of two branches is chosen at random. The first is an application of the simple directed mutation, provided that the segment to shrink is larger than the matched one. The second branch is a union of two adjacent and unmatched segments that are smaller than the one matched, into a single connected segment.

3.4.2. Crossover. The crossover operator for individuals may only be applied when compatible parents are found. Compatibility is defined as the case where each parent has at least

one part matched that is different from any part matched in the other parent. The crossover operation consists of copying any matched parts from either parent into a new individual, then assigning the remaining edges into one of two segment categories based whether one or both parents do not have the edge assigned.

3.5 Algorithm Termination

One of the drawbacks with evolutionary algorithms is that it is difficult to know when evolution may stop. For this reason, the termination criteria must be decided based on the problem domain. The criteria could be a given number of symbols matched if the number is known, as in a zip code or phone number.

In cases where the symbols are joined by extra, non-symbol lines, it is appropriate to define a termination criteria related to the second level mutation. If both branches of the second level mutation fail because appropriate segments can not be found, evolution may be terminated because there is little chance another symbol will be found.

However, this criteria is not appropriate when symbols are touching, or share lines since the unmatched groups tend to get broken where the symbols touch. This commonly results in premature termination, since an actual symbol is broken into two smaller ones.

4. Results

The first example begins with the string presented in figure 1(a) and continues from the individual presented in figure 2(a). Evolution progresses until segments B and C in figure 2(a), are merged in some individual. If not part of the same hypothesis, crossover collects both accepted results into a single, optimal solution.

In some cases, the separation between the generation of a hypothesis and the evaluation of the hypothesis can introduce some errors. These errors appear as over-segmentation, when one accepted symbol may be only a component of the intended symbol. If not accounted for, this may cause the algorithm to terminate prematurely as in the following example.

The following example is taken from figure 10 in [4] to illustrate competing segmentation results, as well as over-segmentation. All grid lines and unconnected characters have been removed manually to provide a clean, connected string for segmentation. Shown below in figure 3(a) is the original image.

The thinning, figure 3(b), graphing, and population initialization were performed producing results including and similar to figure 3(c).

Clearly, in figure 3(c) neither segment represents an intended character in this string. In this case, the initialization algorithm created only two segments since there were no edges that were equally close to either east or west search point.

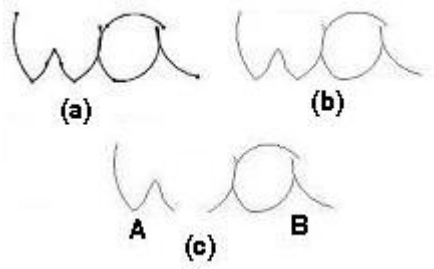

Figure 3. A sample from [4]. (a) The original sample, (b) a thinned version, and (c) an initial hypothesis.

The first result, shown in figure 4(a), is clearly the result of over-segmentation which was allowed to occur since the symbol recognition was trained for many symbols, including numerical digits and the characters 'w' and 'a' as they appear in the thinned image. The segment A is obviously the character 'w', and segment B is accepted as the digit '1'.

Figure 4. Results of evolution on figure 3. (a) An over-segmented result, and (b) the preferred result.

Another result shown in figure 4(b) was produced by evolution, that also satisfied the termination criteria. In this case, both segments are matched to the correct symbols. These segments A and B, match the segments on which the symbol recognition was trained. The system was successful at finding two valid segmentations of the same string, and also segmenting when one edge must belong to two symbols.

High level information such as context or assignment statistics may now be used to decide which result is most appropriate. In this case, the second result is preferred since it uses much more of the original string than the first result, as

well as the having all matched symbols from the same domain of characters.

At the next level of difficulty comes flow diagrams. These diagrams differ from connected digits or characters since the symbols are often arranged both vertically and horizontally. In addition, the connections between symbols account for a much greater portion of the lines that make the string, than in previous digit and character examples.

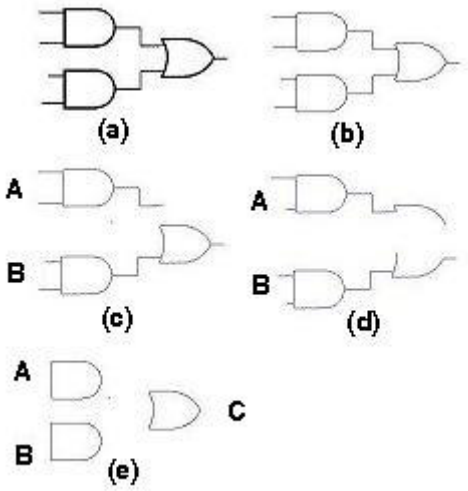

Figure 5. The circuit diagram sample. (a) The original scanned from [7], (b) a thinned version, (c) an initial hypothesis, (d) another initial hypothesis, and (e) the final result.

The same pre-processing is performed, and the individuals in figures 5(c) and 5(d) are created. Mutations may now begin to shrink each one down, in hopes of matching the symbols. The AND gate in figure 5(c) A and 5(d) A can easily be isolated with a few shrinking mutations. The other gates are slightly more difficult, since the second AND gate can only be found through breaking figure 5(c) B or by shrinking 5(d) B. The OR gate must be found by breaking 5(c) B or collecting other segments together.

5. Conclusions.

An evolutionary algorithm provides a number of useful features for solving segmentation problems. The randomness of mutations tends to offset the randomness of the way symbols are connected, it allows for preservation of information in a structured solution, and it gives a starting point to do a multi-directional search. These features allow for the solution to be discovered from either end, or from the middle.

The new system also has the advantage of a separation between symbol verification and the generation and evolution of the individuals. This allows the system to be easily upgradeable or retargetable to new symbol domains. However it was found necessary to maintain a different recognition system for each new domain. This is necessary to minimize the chance of over-segmentation. Clearly, if the character '1' was contained in the recognition system for circuit segmentation, this symbol would be found in nearly every vertical line.

6. Future Work

This type of algorithm contains many areas where improvements can be made. New algorithms can be designed for the mutation or crossover operators, that are adaptable for improvement in a specific domain.

Also, the recognition system implemented only contains a minimal database of symbol models. This made it necessary to manually examine individuals to determine if real symbols are not being matched.

7. References

[1] M. Ahmed and R.Ward, "An Expert System for General Symbol Recognition", *Pattern Recognition 33(12),* 2000, pp 1975-1998.

[2] N.W. Strathy, and C.Y. Suen., "A New System for Reading Handwritten Zip Codes", *Proc. 3rd International Conference on Document Analysis and Recognition,* 1995, pp. 74-77.

[3] M. Parizeau, R. Plamondon, "A Handwriting Model for Syntactic Recognition of Cursive Script", *Proc. of the 11th International Conference on Pattern Recognition (ICPR),* Vol. 2, 1992, pp 308-312.

[4] J.F. Hebert, M.Parizeau, and N. Ghazzali, "Learning to Segment Cursive Words using Isolated Characters", *Proc. of the Vision Interface Conference*, 1999, pp. 33-40.

[5] Y. Yu, A. Samal, and S.C. Seth "A System for Recognizing a Large Class of Engineering Drawings" *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 4. 1997, pp 868 - 890.

[6] M. Ahmed and R. Ward, "A Rotation Invariant Rule-Based Thinning Algorithm for Character Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 12, 2002.

[7] Floyd, T., *Digital Fundamentals 6th Edition*, Prentice-Hal Inc., Upper Saddle River, New Jersey, USA, 1997.