ON THE COMPLEXITY OF COGNITION

S. JAEGER
Interactive Systems Laboratories
University of Karlsruhe
Computer Science Department, 76128 Karlsruhe, Germany
email: stefan.jaeger@ira.uka.de

This paper presents an investigation of a cognitive problem in terms of complexity theory. Two global optimization approaches are presented for recovering trajectories from static, handwritten word images. Both take graph-theoretical representations of symbols as input. The first, polynomial approach minimizes the length of the recovered trajectories. This approach cannot recover trajectories traversing parts of the word more than twice. The second approach, which minimizes costs at distinguished nodes of the trajectory, is more powerful in this respect and is proved to be NP-hard. An efficient divide-and-conquer method is proposed that splits handwritten words into independent subparts and recovers trajectories for every subpart, which turn out to be very small in practice. The splitting technique exploits morphological features from the static word image.

1 Introduction

This paper describes a theoretical investigation towards recovering trajectories from static word images. The solution to this problem would be a major step towards a unified view on handwriting recognition that combines on-line as well as off-line aspects [3, 9, 12]. There is no doubt that recovering trajectories is a difficult problem that touches aspects of many fields such as perception, cognition, and many more. This paper introduces an approach trying to measure this difficulty in terms of complexity theory.

Figure 1 shows a scanned binary image of the letter "B" and its skeleton. The skeleton, which is shown on the right-hand side of Figure 1, is the graph-theoretical abstraction of the scanned word image. A skeleton is a graph comprising nodes and edges, where nodes occur at crossing points or ends of strokes. It coincides in most areas with the centerline of the writing trace and provides us with the necessary formalism for theoretical investigations. In Figure 1, the original trajectory corresponds to the following sequence of edges: $A, B, C, B, D, E, F, G, H$, where the edges C,B, and E must be retraced.

Sections 2 and 3 present two graph-theoretical methods for recovering the original trajectories from skeletons. While the first approach presumes that the original trajectory is a path with the minimum length, the second approach tries to minimize the deviation from the straight writing line. Both methods are global optimization techniques based on the assumption that the perception

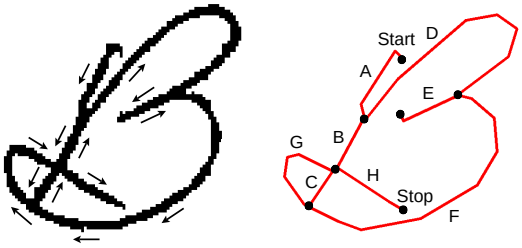

Figure 1: A binary image of the letter "B" and its skeleton.

(or cognition) of handwritten symbols is guided by some (unknown) economical principle.

Sections 2 and 3 address computational complexity issues. Section 4 presents an efficient divide-and-conquer technique that split words into smaller subparts. This method considerably reduces the size of handwritten units in practice.

This paper mainly deals with theoretical issues. However, Reference [9] presents practical evaluations of both methods. Readers not interested in the theoretical issues may skip the theoretical parts of this paper and turn to Section 4, which describes the practical splitting technique.

2 Minimizing Length

This section formulates the recovery of trajectories as a global, graph-theoretical optimization problem. It presumes that the original trajectory is equivalent to the path that has the minimum length and visits every edge at least once. The search for this path is reduced to well-known graph-theoretical methods. In [1], this approach has already been applied to the Arabic language.

Complexity of Length Minimization:

In graph-theoretical terms, the shortest path in a graph that visits every edge at least once is called the Chinese postman path. Accordingly, the search for a Chinese postman path is called the Chinese postman problem [7]. Solving the Chinese postman problem requires the following two main processing steps:

1. Searching for a minimum perfect matching:

   Given a graph $G = [X, U]$ with a set of nodes X and a set of edges U, a *matching* is a subset M of U such that no two edges of M are adjacent [7].

In a perfect matching, every node of G is an end node of at least one edge in M. The word "minimum" requires that the sum of weights in M is minimal. Reference [4] describes an algorithm that finds minimum perfect matchings in polynomial time.

2. Searching for an Eulerian path:

   An Eulerian path is a path that contains every edge of a given graph $G = [X, U]$ exactly once: Searching for Eulerian paths needs only polynomial time [7].

Since both processing steps need polynomial time, solving the Chinese postman problem, i.e. minimizing lengths in skeletons of written symbols, also needs polynomial time. As we will show in the next section, this does not hold for minimizing deviations.

A Chinese postman path has an interesting property: It traverses every edge at most twice. Hence, a Chinese postman path can not describe trajectories that traverse edges more than twice. Reference [9] contains some typical examples of trajectories traversing edges more than twice. For instance, the original writing trace of the "k" shown in Figure 2 corresponds to the following sequence of edges in its skeleton: $A, B, C, B, D, B, E, F, G$. We can see that

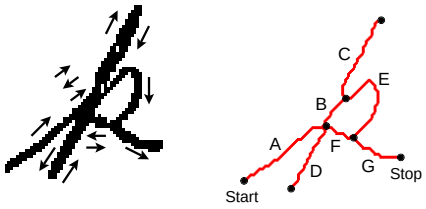

Figure 2: Recovering the trajectory of a "k".

Edge B is traversed three times.

The approach presented in the next section, i.e. minimizing deviations, can describe trajectories that traverse edges n-times. However, we will see that the theoretical complexity of this approach is higher.

3 Minimizing Deviations

This section formulates the recovery of trajectories as a global search for the path with the minimum cost, where costs are deviations from the straight writing line at nodes of the skeleton. It constructs an appropriate problem

representation that allows adequate assignments of costs and a graph theoretical solution to the minimization problem. The new representation is called line graph and is defined as follows:

The *line graph* $L(G)$ of a graph G is a simple graph whose nodes are the edges of G, with two nodes of $L(G)$ adjacent whenever the corresponding edges of G are adjacent [7, 8].

The concept of the line graph has been given different names by different authors, e.g. *interchange graph* or *derivative*, and the reader interested in the properties and characterizations of line graphs is referred to [8]. In Figure 3, the Line graph B_L is superimposed on the skeleton B_S of Figure 1. Consequently,

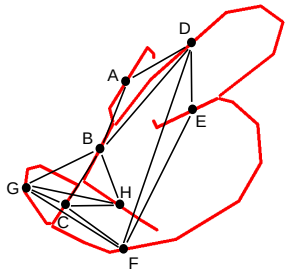

Figure 3: A skeleton of the letter "B" and its line graph.

the nodes of the line graph are given the same labels as the corresponding edges of the skeleton. The line graph provides an appropriate structure for assigning costs of transitions between edges of the skeleton. However, the cost of an immediate and reversed traversal of the same edge, e.g. the traversal of Edge C in B_S, cannot be assigned to an edge of the line graph. In fact, these costs are implicitly set to zero. Figure 4 shows the line graph B_L with symbolic costs assigned to every edge. For instance, $c(A, B)$ is the cost of the transition from Edge A to Edge B in the skeleton of "B". To understand the following processing steps it is sufficient to assume that costs are deviations between edges in the skeleton, which are measured in the vicinity of nodes. In real applications, however, a reliable and precise computation of angles is crucial. Approximated skeletons can help detecting minor changes in directions at nodes. Additionally, local stroke width is useful to determine the vicinity for a node [9].

In order to formalize the recovery of trajectories, we first need to mention the following graph-theoretical definitions:

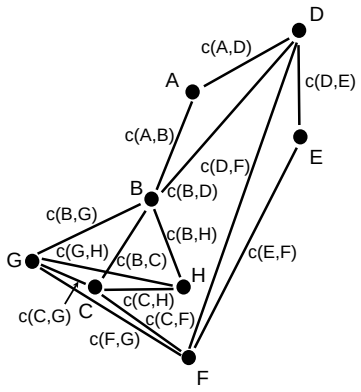

Figure 4: A line graph of the letter "B" with costs assigned to every edge.

A path (or cycle) passing through every node of a graph G **exactly** once is called a *Hamiltonian path* or (*cycle*). A graph G containing a Hamiltonian cycle is called a *Hamiltonian graph*. The search for the shortest Hamiltonian cycle in a graph G is referred to as the *Hamiltonian problem* here.

A path (or cycle) passing through every node of a graph G **at least** once is called a *pre-Hamiltonian path* (or *cycle*) [7].

Using these definitions, the recovery of writing traces is formulated as follows:

Find the pre-Hamiltonian path that has the minimum cost in the line graph.

This problem will be referred to as the *pre-Hamiltonian problem.*

The pre-Hamiltonian problem resembles the Hamiltonian problem. In fact, the pre-Hamiltonian problem in a graph G is identical to the Hamiltonian problem in a complete graph G', where G' is called the *completion* of G [7]. The completion $G'$is a complete graph that has the same set of nodes as G. The weight of an edge (i, j) in G' is the length of the shortest path from i to j in G. To illustrate the relationship between the pre-Hamiltonian problem and the Hamiltonian problem, let us consider the following pre-Hamiltonian cycle in a graph with eight distinct nodes.

$$(K_1, K_6, K_3, ..., K_7, ..., K_2, ..., K_5, ..., K_8, ..., K_4, ..., K_1)$$

While the first occurrence of each node is explicitly shown, the other occurrences are represented by dots. Only the first and last nodes, which are identical, are shown twice. The cost of this specific cycle can be minimized by

traversing the shortest path between any pair of successive nodes explicitly shown. Hence, it is sufficient to find the optimal order of the eight nodes shown and use the costs of shortest paths as distances between these nodes. This is equivalent to the Hamiltonian problem.

The completion B_C of "B" is shown in Figure 5. According to the defini-

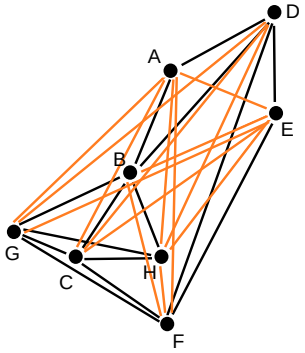

Figure 5: A completed line graph of the letter "B".

tion of completions, every edge (i, j) of B_C is assigned the cost of the shortest path joining i and j in the weighted line graph B_L. For instance, Edge (C, D) represents the shortest path (C, B, D) that joins Node C and Node D in B_L. Consequently, Edge (C, D) is assigned the cost $c(B, C) + c(B, D)$. (The costs are not shown in Figure 5 to keep the figure clear.) Suppose the shortest Hamiltonian path in B_C is (A, B, C, D, E, F, G, H). We can then transform this path into the shortest pre-Hamiltonian path in B_L by expanding Edge (C, D) in (C, B, D). The result is the following path $(A, B, C, B, D, E, F, G, H)$ representing the original trajectory in the skeleton, which contains the retrograde edges B,C, and E [9].

Dijkstra's algorithm [7] computes the shortest paths between all pairs of nodes in B_L with Complexity $O(N^3)$, where N is the number of nodes.

The search for the shortest Hamiltonian cycle in a weighted complete graph is called the *traveling salesman problem* or simply *TSP-problem*. Thus, the approach described here maps the recovery of trajectories to the traveling salesman problem, which is an NP-hard problem.

Reference [9] extends this technique in order to recover paths with the minimum cost that have prescribed beginnings and endings. This is important since the splitting technique proposed in Section 4 requires a prescribed

beginning and ending in order to split words into subparts. Nevertheless, the mapping to the traveling salesman problem is not affected by this extension.

Complexity of Minimizing Deviations

This section shows that minimizing the overall deviation using line graphs is an NP-hard problem. It assumes that the reader is familiar with the terms NP-hard and NP-complete, both central concepts in complexity theory. Readers not familiar with complexity theory can find a comprehensive description of these terms in [5]. According to the state of the art in complexity theory, minimizing deviations is probably computationally intractable, i.e. the existence of a polynomial time algorithm is unlikely. Section 4, however, shows that this problem can be solved in moderate time by dividing the skeleton into subparts. The two main processing steps of minimizing deviations are:

1. generating line graphs from skeletons,

2. searching for the shortest pre-Hamiltonian paths in line graphs.

While the first processing step can be accomplished in polynomial time [7], the second step is NP-hard. Due to the resemblance of the shortest pre-Hamiltonian paths to Hamiltonian paths with minimum costs and the fact that the latter cannot be found in polynomial time, one is tempted to take the NP-hardness for granted. However, the problem is more complicated since we only need to find pre-Hamiltonian paths in line graphs. Moreover, only line graphs that can be derived from *planar* graphs are under consideration.

A graph is *planar* if it can be embedded in the plane, i.e. it can be drawn on a plane surface so that no two edges intersect [8].

If two strokes cross each other, the skeleton necessarily contains a node representing the intersection point. Consequently, every skeleton is planar. The selection of a specific cost function, which assigns costs to every edge of a given line graph, restricts the set of line graphs even further. For this restricted set of graphs the existence of a polynomial time algorithm cannot be excluded in advance. Indeed, in the literature mention is made of solutions in polynomial time for some problems with line graphs. For example, in line graphs the independent set problem, as well as the search for cliques can be solved in polynomial time. The test whether a given graph is a line graph can also be done in polynomial time. More details and references can be found in [11].

The NP-hardness proof proceeds in two steps and shows the following:

- First, that the search for Hamiltonian paths (not pre-Hamiltonian paths) in line graphs derived from planar graphs is NP-hard (hereafter referred to as Problem 1).

- Second, that when costs are introduced, the search for the shortest pre-Hamiltonian paths is NP-hard (hereafter referred to as Problem 2).

Actually, it is the latter problem we are mainly interested in. Note that we will only show the NP-hardness for a small subset of all possible skeletons. In fact, the search for the shortest pre-Hamiltonian paths in planar, cubic, and triply-connected skeletons is be shown to be NP-hard.

Problem 1 can be solved with the help of known research results. In [2], the search for Hamiltonian paths in line graphs is referred to as the *edge Hamiltonian path problem.* The edge Hamiltonian path problem is shown to be NP-hard by reducing the *cubic Hamiltonian Path problem* to it. The cubic Hamiltonian path problem, i.e. the search for Hamiltonian paths in cubic graphs, is a known NP-complete problem. A graph is said to be a *cubic graph* if every node is shared by exactly three edges, i.e. every node has Degree 3. However, cubic graphs need not always be planar. A well-known example of this is shown in Figure 6. To ensure planarity, we confine the set of cubic graphs

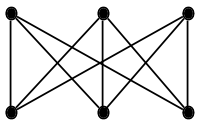

Figure 6: A cubic, nonplanar graph.

used in [2] to planar, cubic graphs. This restriction to planar, cubic graphs has no severe impact on the NP-hardness proof in [2]. This follows from the analysis of [6] which demonstrates that the search for Hamiltonian cycles or paths in planar, cubic, triply-connected graphs is NP-hard. A graph is said to be *triply-connected* if a deletion of any two vertices leaves the graph connected. The evidence for NP-hardness in [6] is provided by reducing the satisfiability problem of propositional calculus, a well known NP-hard problem, to the Hamiltonian problem in planar, cubic, triply-connected graphs. Hence, the proof of [2] remains valid and can be applied to planar, cubic graphs. Consequently, the search for Hamiltonian paths in line graphs derived from planar, cubic graphs remains NP-hard (Problem 1).

The result of Problem 1 helps solving the second problem, which will provide direct proof that the recovery of writing traces, as it is understood here,

is NP-hard. Of course, the NP-hardness proof strongly depends on the costs we assign to every edge of the line graph. This paper shows NP-hardness for the definition of costs as it is shown in Figure 7. Suppose a writer has started at

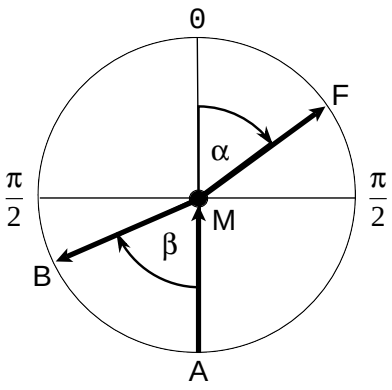

Figure 7: The definition of deviations.

Point A in Figure 7 and has reached the center of the circle M. If he is heading toward Point F, the cost is defined by Angle α. If he is heading toward Point B, the cost is defined by Angle β. Thus, all deviations are measured in the range from $0°(0)$ to $90°(\frac{\pi}{2})$. Naturally, the cost of a 90°-angle is higher than the cost of a 0°-angle, which is in accordance with the fact that straight lines are easier to write than right angles. The deviation between a straight line and its retraced stroke is zero. Note that the cost of a trajectory is independent of the direction in which it is traversed. This symmetry is an intrinsic characteristic of the definition of costs shown in Figure 7. Reference [9] shows that using an alternative cost function that assigns high costs to retrograde strokes does not avoid NP-hardness.

The proof of Problem 2 exploits the solution to Problem 1, i.e. the fact that searching for Hamiltonian paths in line graphs derived from planar, cubic graphs is NP-hard. It shows that a polynomial solution to Problem 2 would entail a polynomial solution to Problem 1, which is a contradiction. Since the proof of Problem 2 is technical (see [9] for more details), we only present the following theorem as the final result:

Theorem: The search for the shortest pre-Hamiltonian paths in line graphs derived from skeletons of written symbols is NP-hard.

4 Splitting Technique

While searching for a minimum perfect matching is the most time-consuming part of the Chinese postman problem, solving the traveling salesman problem requires an expensive search for a traveling salesman tour. Since the practical requirements are very stringent, e.g. throughput requirements in postal automation, it is essential for the searches to be efficient. This is accomplished by splitting words into subparts. Skeletons of words often comprise several distinct, connected components which are independent from each other. A first splitting step is to recover trajectories independently for each connected component. However, if there is only one connected component or the exisiting components still contain too many edges, it may be necessary to reduce the number of nodes further. This section shows an efficient divide-and-conquer technique detecting splitting edges in the skeleton that are traversed only once and from left to right. Splitting edges divide the skeleton into distinct parts in which trajectories can be recovered independently. They meet two requirements:

1. A splitting edge is not part of the outline of an enclosure.
   (Reference [13] describes an efficient algorithm that detects enclosures and verifies this feature for off-line images.)

2. A splitting edge has two distinct nodes. One node is nearer to the beginning of the trajectory and the other one is nearer to the end, where a node p is nearer to the beginning (end) than a node p' if the shortest path joining p to the beginning (end) is shorter than any other path joining p' to the beginning (end).

Figure 8 shows the decomposition of a skeleton. The skeleton consists of two

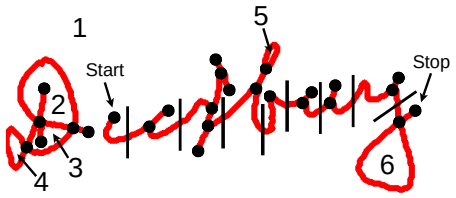

Figure 8: The decomposition of a skeleton.

connected components and contains five enclosures, each having been assigned a number in Figure 8. Since algorithm [13] regards the writing pad, i.e. background, as a special type of enclosure, the numbering ranges from 1 to 6. The

right-hand connected component is decomposed into nine components using eight splitting edges. Note that splitting edges can divide a single character into two or more components. Experimental tests with handwritten American city names show an average of approximately four edges per component. Components containing more than sixty edges may occur. Nevertheless, these components are very rare.

A disadvantage of the proposed method is that it needs a prescribed beginning and ending of the trajectory. For this reason, a simple heuristic, which is based on pairs of potential candidates for beginnings and endings, is given in [9].

5 Result and Conclusion

This paper describes the hardness of a specific cognitive problem in terms of complexity theory. Two different graph-theoretical approaches to recovering trajectories from static word images have been presented, both global optimizations. The first approach, which minimizes the length of the recovered trajectory, has polynomial complexity but cannot recover trajectories that traverse parts of the word more than twice. The second approach, which minimizes the deviation from the straight writing line, is more powerful in this respect. However, it is proved to be NP-hard. An efficient divide-and-conquer technique has been presented that applies both approaches to smaller subparts to improve efficiency. This technique has no effect on the finally recovered trajectory.

A drawback of both approaches is that they cannot recover trajectories containing several pen-ups and pen-downs. Nevertheless, it is possible to extend both approaches to model pen-ups and pen-downs [9]. This has not been investigated further since the consistent traversal of letters, which is the most important requirement for character recognition, is not necessarily affected by this restriction.

The average edit distance between original and recovered trajectories, i.e. the number of insertions and deletions of edges, is about 4 for the traveling salesman approach. This distance was measured on an on-line database containing 7000 words, where the average length of a trajectory is 17. The TSP approach performs only slightly better than the Chinese postman approach on this database. Reference [9] presents some successfully recovered trajectories.

In another experiment, the on-line recognition system NPen++ [10] was trained and tested with recovered on-line information extracted from addresses containing handwritten American city names (TSP approach). Though the genuine off-line recognition rates are still higher (94% on a 400 word dictionary), a recognition rate about 73% was reached with recovered on-line data (see [9] for more details).

Acknowledgments

Substantial parts of this work were carried out at the Daimler-Benz Research Center in Ulm, Germany. I wish to thank Jürgen Schürmann, Norbert Bartneck, Bernhard Nebel, and Stefan Manke for their support. Many thanks also to Eberhard Mandler and Ulrich Kressel for the fruitful discussions we had.

References

1. I. S. Abuhaiba and P. Ahmed. Restoration of Temporal Information in Off-line Arabic Handwriting. *Pattern Recognition*, 26(7):1009–1017, 1993.
2. A. A. Bertossi. The Edge Hamiltonian Path Problem is NP-Complete. *Information Processing Letters*, 13(4,5):157–159, 1981.
3. G. Boccignone, A. Chianese, L. P. Cordella, and A. Marcelli. Recovering Dynamic Information from Static Handwriting. *Pattern Recognition*, 26(3):409–418, 1993.
4. J. Edmonds and E. L. Johnson. Matching, Euler Tours and the Chinese Postman. *Mathematical Programming*, (5):88–124, 1973.
5. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. H. Freeman & Co., New York, 1979.
6. M. R. Garey, D. S. Johnson, and R. E. Tarjan. The Planar Hamiltonian Circuit Problem is NP-Complete. *SIAM J. Comput.*, 5(4):704–714, 1976.
7. M. Gondran and M. Minoux. *Graphs and Algorithms*. Wiley, 1984.
8. F. Harary. *Graph Theory*. Addison-Wesley, Reading Mass., 1969.
9. S. Jaeger. *Recovering Dynamic Information from Static, Handwritten Word Images*. PhD thesis, University of Freiburg, 1998. Foelbach Verlag.
10. S. Jaeger, S. Manke, and A. Waibel. Npen++: An On-Line Handwriting Recognition System. In *7th International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, Amsterdam, 2000.
11. D. S. Johnson. The NP-Completeness Column: an Ongoing Guide. *Journal of Algorithms*, pages 434–451, 1985.
12. P. M. Lallican and C. Viard-Gaudin. A Kalman Approach for Stroke Order Recovering from Off-line Handwriting. In *Fourth International Conference on Document Analysis and Recognition (ICDAR)*, pages 519–522, 1997.
13. E. Mandler and M. F. Oberländer. A Single Pass Algorithm for Fast Contour Coding of Binary Images (in german). In *Proc. of the 12th DAGM-Symposium*, pages 248–255, 1990.