3 Polygonal approximation

3.1 Introduction

3.1.1 Problem formulation

The general problem of approximation a given two-dimensional piecewise linear curve by another coarser one is of fundamental importance in computer graphics, vectorization tasks, vector map processing (see Figs. 3.1.3–3.1.7).

An open *N-vertex polygonal curve P* in 2-dimensional space is represented as the ordered set of vertices $P=\{p_1, \ldots, p_N\}=\{(x_1, y_1), \ldots, (x_N, y_N)\}$. The output coarser curve Q consists of $(M+1)$ vertices: $Q=\{q_1, \ldots, q_{M+1}\}$, where the set of vertices q_m is a subset of P and $M<N$. The end points of Q are the end points of P: $q_1 = p_1$, $q_{M+1} = p_N$. The approximation linear segment (q_m, q_{m+1}) of Q for curve segment $\{p_i, \ldots, p_j\}$ of P is defined by the end points p_i and p_j: $q_m = p_i$ and $q_{m+1} = p_j$. Thus, $(q_m, q_{m+1}) = (p_i, p_j)$.

As it was stated in [147, 119] there are two types of optimization problems connected with polygonal approximation problems:

Min-ε problem: Given a polygonal curve P, approximate it by another polygonal curve Q with a given number of line segments M so that the approximation error $E(P)$ is minimized.

Min-# problem: Given a polygonal curve P, approximate it by another polygonal curve Q with the minimum number of segments M so that the approximation error $E(P)$ does not exceed a given maximum tolerance ε.

3.1.2 Error measures

An approximation curve must satisfy some error criterion, which is specified appropriately for each application. In practice, the most of practical error measures

in use are based on distance between vertices of the input curve and the approximation linear segments.

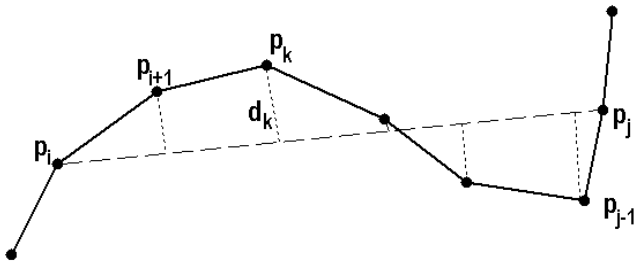

Figure 3.1.1: Distance d_k from the point p_k to the linear segment (p_i, p_j).

The distance $d_k(i,j)$ from curve vertex $p_k=(x_k, y_k)$ to the corresponding approximation linear segments (p_i, p_j) is defined as follows (see Fig. 3.1.1):

$$d(k;i,j) = \frac{| y_k - a_{i,j}x_k - b_{i,j} |}{\sqrt{1 + a_{i,j}^2}} \tag{3.1.1}$$

where the coefficients $a_{i,j}$ and $b_{i,j}$ are defined from the parameters of the linear segment (p_i, p_j):

$$\begin{aligned} a_{i,j} &= (y_j - y_i)/(x_j - x_i) \\ b_{i,j} &= y_i - a_{i,j}x_i. \end{aligned} \tag{3.1.2}$$

The additive error measure L_p for curve segment $\{p_i, p_j\}$ is defined by the sum of distances $d(k; i, j)$ for all vertices in the segment as follows:

$$e_p(i,j) = \sum_{k=i+1}^{k=j-1} d^p(k;i,j) \tag{3.1.3}$$

For L_∞ the approximation error is defiend as the maximum deviation of input curves from approximation linear segment:

$$e_\infty(i,j) = \max_{i<k<j}\{d(k;i,j)\} \tag{3.1.4}$$

Approximation error $E_p(P)$ of the input curve P by Q with additive error measure L_p (where $p<\infty$) is defined as the sum of approximation errors for all segments:

$$E_p(P) = \sum_{m=1}^{M} e_p(i,j) \tag{3.1.5}$$

For error measure L_∞, the approximation error for the curve P is defined as the maximum of distances for all segments:

$$\Delta(P) = \max_{1 \le m \le M} \{e_\infty(i, j)\} \tag{3.1.6}$$

The error measure L_2 (integral square error, or ISE) is perhaps among the most widely used criteria for approximation *min-ε problem*. The error $e_2(p_i, p_j)$ with measure L_2 can be calculated in $O(1)$ time using stored arrays of coordinates cumulatives.

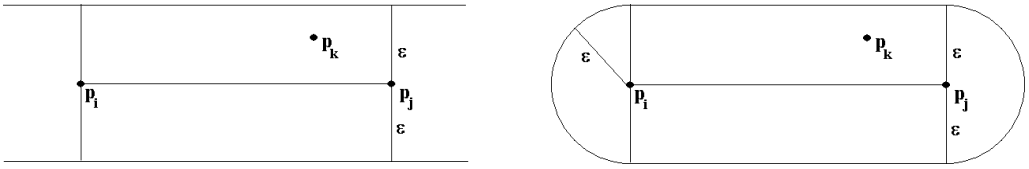

Figure 3.1.2. Error criteria with measure L_∞: the *parallel-strip* (or *infinite beam*) criterion (left), and the *segment distance* (or *tolerance zone*) criterion (right).

In most cases the error measure L_∞ is used in algorithms for *min-# problem* for practical reasons. In [119] different error criteria with measure L_∞ are given, including the most popular *parallel-strip* (or *infinite beam*) criterion [269], that is maximum distance between the line connecting p_i and p_j and the points of the curve segment $\{p_i, ..., p_j\}$ (see Fig. 3.1.2, left). The *segment distance* (or *tolerance zone*) criterion is defined as maximum distance between the line segment (p_i, p_j) and the points of the curve segment $\{p_i, ..., p_j\}$ (see Fig. 3.1.2, right). In some algorithms, other error measures are in use for *min-# problem*: integral square error L_2 [233, 243], and local integral square error (LISE) [52], or more complicated objective functions [212]. However, using error measure L_2 for *min-# problem* is not efficient in practice, especially in the case of multiple objects, because of its additive nature.

Since optimal algorithms are computationally expensive to be used (usually between $O(N^2)$ and $O(N^3)$), faster sub-optimal algorithms have been developed, often running in linear time. In order to evaluate the quality of sub-optimal algorithms, Rosin [224] introduced two measures. *Fidelity* (F) measures how well the suboptimal polygon fits the curve relative to the optimal polygon in terms of the approximation error. *Efficiency* measures how compact the suboptimal polygonal presentation of the curve is. They are defined as follows:

$$\text{Fidelity} = \frac{E_{\min}}{E} \times 100$$

$$\text{Efficiency} = \frac{M_{\min}}{M} \times 100$$

where M is the number of segments for an algorithm under question, $M_{\min}$ is the number of segments for *min*-# solution, E is approximation error for an approximation algorithm, and $E_{\min}$ is the approximation error of the optimal solution for the *min*-ε *problem*.

3.1.3 Motivation

Important problems such as polygonal approximation have been explored very intensively for the last 30 years. The *min*-# and *min*-ε problems can be solved as an optimization task by dynamic programming or methods based on graph theory. Moreover, many heuristic algorithms have proposed. Optimal algorithms of complexity $O(N^2)$–$O(N^3)$ are very slow, whereas faster heuristic algorithms lack of optimality. Thus, development of *efficient* (fast and optimal or near-optimal) algorithms for large input data is still an open problem.

Introducing into practice more efficient algorithms for *min*-# problem we can reduce storage demands or transmission time for the same tolerance level. With more efficient algorithms for *min*-ε problem, we can reduce approximation error for the same amount of stored or transmitted data. Taking into account that nowadays polygonal approximation in vectorization, map service, CAD and GIS applications, the efficient solution of this problem is still of great practical importance (see Figs. 3.1.3-3.1.7).

At first we provide a short survey of heuristic algorithms for *min*-# and *min*-ε approximation problems in Section 3.2. Then we explore optimal algorithms for *min*-ε *problems* in Section 3.3. To bridge the gap between slow optimal and fast heuristic non-optimal algorithms we introduce paradigms of *bounding corridor* and *iterated reduced search* [**P4**]. Then we study optimal algorithms for *min*-# problem and present algorithm with joint using of different error measures based on the reduced search [**P5**]. Furthermore, we consider *min*-ε and *min*-# *problems* for the case of *closed contours* and provide cyclical DP algorithm with analysis of the state space [**P6**]. Finally, we extend the proposed iterative reduced search approach to the case of *min*-ε *problem* for *multiple objects* [**P7**].

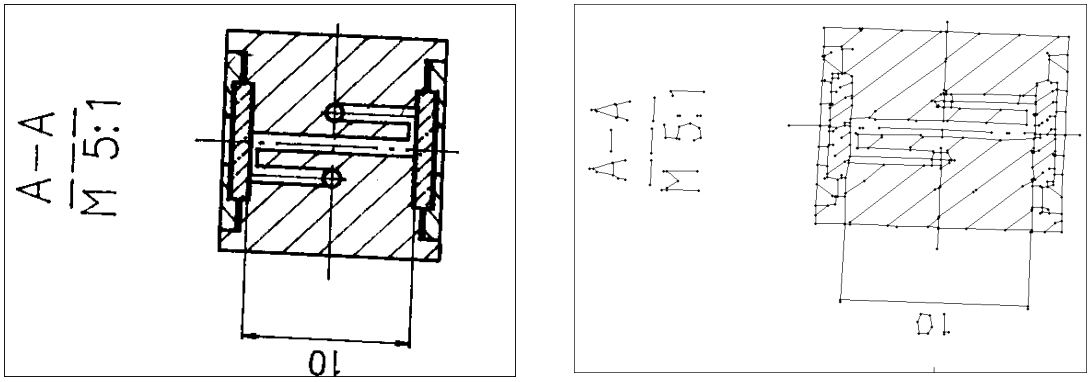

Figure 3.1.3: *Min*-# approximation of digitized curves (skeletons) for raster-to-vector conversion: input raster binary image (left); vector image for error tolerance ε=1.5 (right). Vectors are labeled by dots.

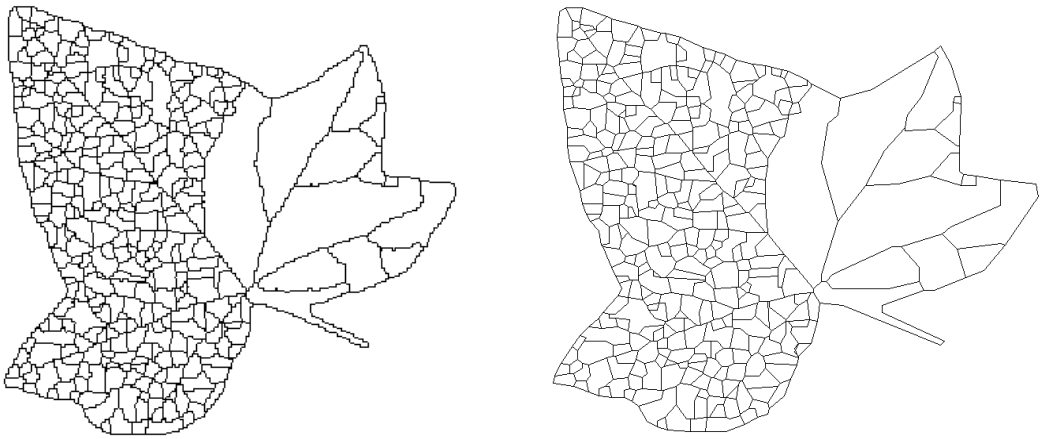

Figure 3.1.4: *Min*-# approximation of digitized curves: segmentation data (left); result of approximation for error tolerance ε=1.5 (right).

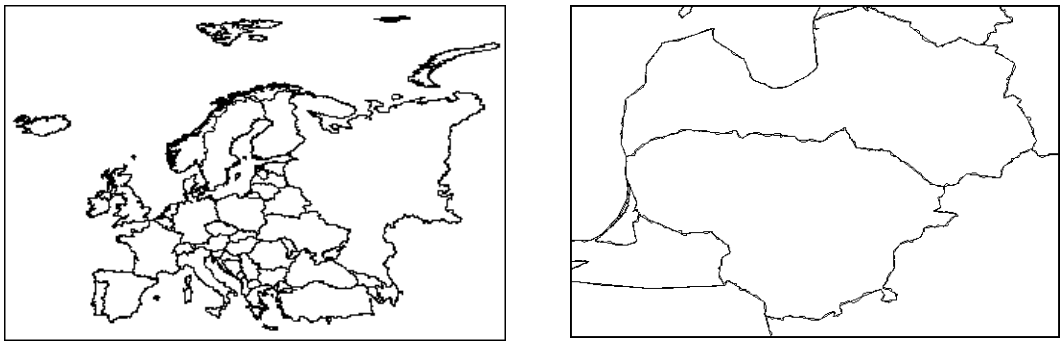

Figure 3.1.5: *Min*-ε approximation of multiple-object vector data: vector map of "Europe"; 365 objects with 160,000 vertices (left); fragment of the vector map after 20:1 data reduction (right).

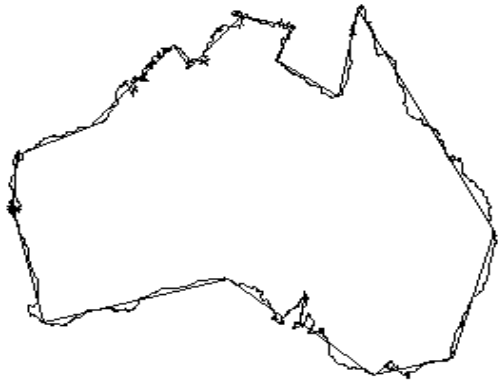

Figure 3.1.6. *Min*-ε approximation of closed 2900-vertex contour “Australia” by *M*=18 linear segments.

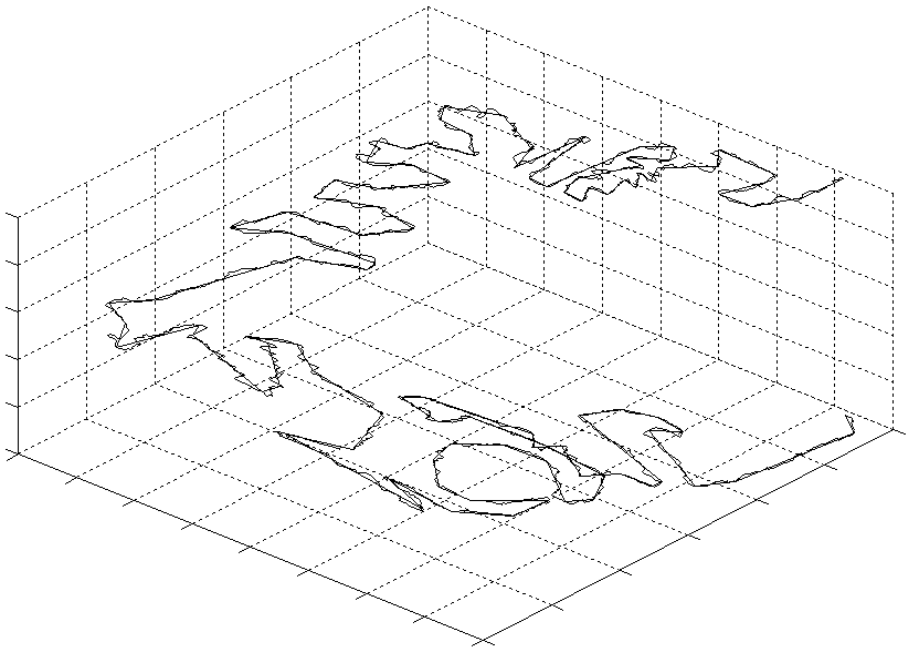

Figure 3.1.7: *Min*-ε approximation of 5000-vertex digitized path in 3-D space by *M*=100 linear segments.

3.2 Heuristic algorithms

3.2.1 Survey of solutions

For the last 30 years many heuristic algorithms have been considered for approximation of polygonal curves. We can account about dozen of different heuristic approaches to the problem, and the number of algorithms exceeded one hundred items. In some extent, the existence of big amount and variety of the heuristic approximation algorithms can be explained by the variety of tasks, curves types, and error measures in use. But more likely the real reason for existence of numerous approximation methods is low fidelity and/or efficiency of the proposed heuristic algorithms [224, 225], which leaves room for improvement.

Let us briefly consider the following approaches proposed for solving approximation problems. Some of the presented methods can be used for solving both problems, but some of the algorithms are designed for *min*-# or *min*-ε problem only. With algorithm for one of the problems (*min*-# or *min*-e) we can get solution for alternative problem in $O(\log N)$ steps of binary search [37].

a) Sequential tracing approach

The algorithms [249, 216, 290, 250, 147, 94, 121, 284, 218, 54, 12, 213, 143, 316] use a linear scan to evaluate error conditions, if the conditions are not satisfied a new segment search is started. The main problem of the methods is that the nodes sometimes do not correspond to the corners of the curve because a new vector is defined only when the criterion is violated.

b) Split method

The most widely used high-quality approximation algorithm is a heuristic method called the Douglas-Peucker algorithm [71]. It has been independently invented by many people [207, 72, 24, 21]. The iterative procedure repeatedly splits the curve into smaller and smaller curves until the maximum of the perpendicular distances of the points on the curve from the line segment is smaller than the error tolerance ε. The complexity of the method is $O(N^2)$ in the worst case, and $O(N\log N)$ on average. The algorithm works in any dimension since it only depends on computing the

distance between points and lines. The main disadvantage of this approach is the dependency on the starting point. It also suffers from stressing outliers, see more critics in [279, 280]. Modified version of Douglas-Peucker algorithm was proposed [104, 105] complexity of O(N log$N$) in the worst case, based on construction of convex hull of 2D point set. Later the result has been improved to $O(N \log^* N)$ [106]. Unfortunately, the faster algorithm is not general, as it only works with simple 2-D planar curves, and not in higher dimensions.

c) Merge method

A common idea in some algorithms such as Sequential Tracing, Split, Dominant point detection, is to choose curve points to be vertices of the polygonal approximation. It can be done in opposite direction by choosing, at each stage, a curve point that will not be a vertex [157, 78, 294, 33, 281, 146, 198, 112, 150]. A reasonable choice is a point of which elimination will cause minimal increase in the approximation error. The procedure is halted when the desired number of linear segments M, or approximation error, is reached. Result is independent on the starting point. The complexity of the algorithm by Pikaz and Dinstein [198] is $O(N \log N)$.

d) Split-and-Merge method

According to the technique, lines are fitted to an initial segmentation of the boundary and the least squares is computed. The procedure then iteratively splits a line if the error is too large and merges two points if the error is too small [192, 10, 294, 189, 103, 215, 300]. This combines the split and merge methods with the same algorithm.

e) Dominant point detection

Attneave [18] indicates that most shape information is contained in the corners (high curvature points), which are able to characterize the contour. To approximate curves using straight lines, high curvature points are the best place at which to break the lines. A large number of heuristic algorithms have been designed on the basis of the idea [220, 223, 86, 59, 236, 35, 9, 284, 17, 79, 263, 62, 10, 76, 299, 210, 211, 237, 296, 14, 319, 56, 115, 89, 120, 234, 235, 251, 23, 173, 293, 162].

f) Relaxation labeling

According to the approach, the left and right slopes and curvature value is measured at every point of the input digital curve. Each point on the curve is associated with an attribute list containing slopes and curvature at the point. The attributes will determine the initial *probability* of the point p_i being a ‘side, and probability of

being an ‘angle’. The relaxation process will change the probabilities. As the relaxation process is iterated, certain points became more certain that they are ‘angles’, while other points became more certain that they are ‘side’. Finally, the probabilities converge to some values [60, 61, 227, 166].

g) *K*-means method

Phillips and Rosenfeld [196] proposed *k*-means based clustering method to partition the contours into subsets of points such that each subset can be fitted by a straight line. They started with an initial partition and then evaluate the principal axis. In [304] three algorithms have been proposed based on Phillips and Rosenfeld approach for *min*-ε *problem* with two error measures (L_2 and L_∞). A simpler line fitting method was used instead of the principal axis approach.

i) Genetic (evolutional) algorithms

Genetic algorithms [305-307, 116, 215, 63, 259, 270, 108, 312] are based on stochastic search, which simulates the biological model of evolution [93]. A population is set of chromosomes and an initial one is randomly generated. During each generation, the fitness (approximation error) of each chromosome is evaluated, and chromosomes are selected for reproduction based on their fitness values. Selection operator reproduces some chromosomes with smaller approximation error and eliminates chromosomes with bad approximation. The selected solutions the undergo reproduction under action of the crossover and mutation operations.

j) Ant colony optimization method

Ant Colony Optimization (ACO) is an optimization paradigm that mimics the exploration strategy of a colony of ants [67]: ants can construct the shortest path from their colony to the feeding source and back using pheromone trails. An ant leaves some quantities of pheromone on the ground and marks the path by a trail of this substances. The next ant choose path depending on the amount of pheromone on it and leaves its own pheromone. Vallone [277] considered *min*-ε *problem* with maximum perimeter of polygon as an optimization criteria. Yin [308] considered *min*-# *problem* with error metrics L_2, including approximation of closed contours.

k) Tabu search

Tabu search, developed by Glover [91], is one of the meta-heuristic methods that can be used to solve combinatorial optimization problems. It is different from the local search in the sense that tabu search allows moves to a new solution, which makes the objective function worse in hope that it will achieve a better solution in a longer term [Yin'00, Zhang-Guo'01].

l) Vertex adjustment method

The main idea of the *vertex adjustment method* is to improve preliminary result by a local search [44, 132, 153, 49, 311, 110, 173]. The local search can be applied to vertices successively (vertex-by-vertex) [132, 49], or simultaneously to all vertices using optimization technique [44, 153, 110]. Solution of the optimization task can be found by Viterbi algorithm for the shortest path in a graph [153] or by dynamic programming [44, 110, 173].

In [153], the adjustment of approximation points is performed by Viterbi algorithm for the shortest path in graph. The search is performed among 4-neigbours of the initial approximation points. This technique, however, can be extended to the case where approximation points belong the input curve only.

Chen *et al.* [44] studied *min*-ε *problem* for approximation by circular arcs and line segments they introduced modification to the dynamic programming algorithm for the problem in question. To reduce processing time, they search all possible combinations of the points that are within a given range with respect to an initial set of detected break points, instead of performing complete enumeration. The initial set of break points, $\{u_i,\ i=1, 2, ..., N\}$, is the solution obtained with heuristic algorithm for dominant (break) point detection [44]. The possible sets of solutions are generated by varying each u_i within a range of (u_{i-h}, u_{i+h}), where $h=3$. Although this method has been proposed for approximation by circular arcs and lines, the approach can be applied in the case of piecewise linear approximation as well.

In [153] the adjustment of approximation points is performed by Viterbi algorithm for the shortest path in graph. The search is performed among 4-neigbours of the initial approximation points. However this technique can be extended to the case when approximation points belong to the input curve only.

Neumann and Teisseron [173] and Horng [110] used the same approach as Chen *et al*. [44] to reduce the processing time of dynamic programming algorithm. They performed search of the optimal location of approximation vertices within given range around the current position. Neumann and Teisseron defined the window where the dominant point can be located. Horng defined the window size as 1/3 of the number of the vertices in the curve segment between two correspondent

dominant points. In both cases, the set of the initial approximation points $\{q_1, \ldots, q_{M+1}\}$ is defined by algorithms for the dominant points detection.

3.2.2 Summary

The presented heuristic approaches can be divided into two classes from optimality point of view:

1) Classical algorithms, namely sequential algorithms, split, merge, split-and-merge, dominant points detection, relaxation labeling;

2) Optimization algorithms: *K*-means, tabu search, genetic algorithms, ant colony optimization methods, and vertex adjustment methods.

The classical algorithms are mostly based on heuristic methods or approaches. The fidelity of these algorithms is usually low [224] because the global optimization error is not subject to control during the process of the approximation. Some heuristic algorithms can be used as part of optimal algorithms. For example, the cone-intersection method was used in graph theory based algorithms to reduce the complexity of the graph construction procedure. Sallotti [230, 231] applied heuristic (Split) algorithm of Pavlidis [190] to estimate upper bound for approximation error to reduce A*-search in graph.

In optimization algorithms the approximation problem is considered as optimization task where the global approximation error is the main criterion to be controlled. The search of solution that provides minimal approximation error can be performed by *stochastic optimization methods* (as genetic algorithms and ant colony method) or by *local optimization methods* (as tabu search and vertex adjustment methods). The initial solution for starting the search can be obtained with some heuristic algorithm for approximation, or any random approximation can be used. Then the initial approximation (or approximations) is improved to find minimum of the global approximation error. Algorithms of this class can provide near-optimal or sometimes optimal results, but the global optimality cannot be guaranteed even in the case of iterative approaches.

3.3 Optimal algorithms for *min*-ε *problem*

The min-ε *problem* is formulated as follows: given polygonal curve P, approximate it by another polygonal curve Q with a given number of segments the minimum number of segments M so that the approximation error is minimal.

3.3.1. Early history of the subject

The early history of *min*-ε *problem* was begun from the problem of L_2-optimal approximation for continuous one-variable function. In 1961 Stone [257] considered piecewise-linear curve fitting as a formal optimization problem. The objective was to minimize the squared approximation error subject to a constraint on the number of linear segments. Bellman followed with a solution [27] based on his principle of optimality [26]. Later Gluss [92] expanded upon Bellman's work. Lawson [151] used dynamic programming (DP) to establish the existence of a balanced error property. Cox discussed a similar approach in his paper [57]. Cantoni [34] determined the optimal polygon of a known nonlinear function by minimizing the weighted integral square errors. The works performed by Bellman, Gluss, Stone, Cantoni and Cox hold only for a 1-D signal whose analytic forms are known. Nevertheless, it inspired other researchers to use the dynamic programming approach for optimal approximation of digital curves [194, 44].

3.3.2 Dynamic programming algorithm for *min*-ε *problem*

In 1994 Perez and Vidal published the first optimal algorithm for *min*-ε *problem* for digital curves [194]. The proposed algorithm was based on dynamic programming method for solving optimization task. The authors extended DP approach for approximation of 1D continuous functions [27, 92] to the case of digital 1-D waveforms and 2-D planar curves. They wrote that the approach can be extended on the case of 3-D space as well with corresponding approximation error function. To illustrate their method Perez and Vidal presented solutions for error measures L_1 and L_2. The complexity of optimal algorithm with error measure L_1 is $O(NM^3)$, but complexity of the algorithm for error L_2-norm was reduced to $O(NM^2)$ by using the incremental scheme for computation of approximation error. Authors also considered approximation of closed contours and pointed out that at most (N-M)

runs of the basic algorithms are necessary to find globally optimal solution. They also mentioned cyclical DP as a possible way for solving approximation problem for closed contours.

In 1996, Chen *et al.* [44] studied *min*-ε *problem* with measures L_2 and L_∞ for approximation of 2-D planar curves by circular arcs and line segments they proposed algorithm, which was also inspired by DP method of Bellman [27, 92] for 1-D continuous functions.

Dahl and Realfsen [58] presented *min*-ε problem as searching of shortest path in a directed acyclic graph containing at most M arcs. Regrettably, they restricted consideration by the case $M>N/2$ only. Moreover, the DP search of the approximation error minimum is limited to three neighbouring vertices only. Generally speaking, it makes the algorithm sub-optimal. Anyway, for the case under consideration (large number of segments relatively the number of vertices) the vertex adjustment-based approach can provide results that are quite close to the optimal approximation.

Heckbert and Garland published a survey in 1997 [101], and they wrote about optimal approximation of digital 1-D digital waveform $f(x)$ with minimum error: "*the L_2-optimal approximation to a function f(x) can be found in $O(MN^2)$ time, worst case, using dynamic programming*", but they did not provide any details of the algorithm or references.

Haugland *et al*. [97, 98, 100] represented optimal algorithm for *min*-ε *problem* for 1-D waveforms (ECG) with error measure L_2. In contrast to Perez-Vidal algorithm, the proposed solution was based on algorithm for resource-constrained shortest path in graph, introduced by Saigal [228] and later corrected by Rosseel [226]. Nevertheless, from computational point of view the suggested DP-based algorithm for cardinality constrained shortest path (CCSP) is equivalent to Perez-Vidal algorithm for 1-D signals. The CCSP approximation algorithm was used for lossy compression of ECG signals with a given compression ratio [180, 178, 181]. Furthermore, the CCSP algorithm was extended to the case of planar curves [182, 178]. This DP algorithm is also equivalent to solution originally given by Perez and Vidal [194] for 2-D planar curves.

Tseng *et al*. [274] presented DP algorithm for optimal approximation with a given error tolerance (*min-# problem*). Three error measures were used, including L_1, L_∞ and a length cost function. The essence of the algorithm is the same as that of Perez and Vidal. The only difference is the stop rule: in Perez-Vidal algorithm the DP search in the state space is continued until a given number M of segments is

reached, in the algorithm of Tseng *et al*. the search is performed until the current approximation error is less than a given error tolerance ε.

Mori *et al*. [171] proposed DP algorithm for optimal approximation of curves by linear segments, circular arcs and splines. The approach is the same as that of Perez and Vidal. The DP approach of Perez and Vidal was used for approximation of input curve using circular arcs [193], circular arcs and line segments [111]. In [52] the Perez-Vidal algorithm was extended to the case of polygonal approximation in 3-D space with local integral square error (LISE) measure.

The main contribution to the reduction of the complexity of Perez-Vidal algorithm has been done by Salotti [230, 231]. The main idea behind the algorithm is to stop the search as soon as possible using heuristic functions to estimate the cost function. The algorithm has been implemented in two versions: A*-search [230, 231] and dynamic programming [232]. At first, the rough approximation with Pavlidis algorithm [190] is performed to find approximation error to be used further as upper bound for approximation error γ. Then A* or DP search is performed. If the estimated cost function for the current vertex is bigger than the upper bound γ, the next vertices located further along the curve P are not examined as possible candidates as approximation points of Q. Salotti offered two methods for estimation of the remaining cost function from the current vertex to the goal vertex. The use of heuristics in the algorithms makes it difficult to estimate the complexity. Experiments provided with test shapes have shown that the complexity of the algorithm is close to $O(N^2)$.

The paper of Perez-Vidal [194] is the key publication for *min*-ε *problem* for digital curves because it was the first publication where the optimal algorithm for the problem has been proposed. Moreover, it was the first paper, which contained deep analysis of all the questions concerning the *min*-ε *problem*. In more recent publications [97, 98, 100, 182, 177, 274, 171] the algorithm of Perez and Vidal has been rediscovered.

Min-ε *problem for concave/convex curves*

Chan and Chin [37] proposed L_∞-optimal solution for *min*-ε problem for the convex curves of complexity $O(N^2)$ with algorithm, based on the search of the shortest path in directed graph. They offered to take advantage of the convexity of the input curve P to construct the graph $G(P)$ on the vertices of the P in $O(N^2)$ time, and to find the *min*-ε approximation with an additional $O(MN)$ time.

Aggarwal *et al*. used matrix search algorithm for cost functions with Monge property to solve optimization tasks: *min*-ε *problem* for the closed convex/concave curves with maximum area or perimeter cost function [7] and finding a minimum-weight *k*-link path in graphs [8]. For monotonic cost function with Monge property the optimization problem can be solved by DP algorithm with matrix search in $O(MN)$ time [7, 297, 298]. Thus, in the case of convex/concave curves, the *min*-ε *problem* with L_2 error metrics can be solved with matrix search algorithm in $O(MN)$ time [7]. Regrettably, in most cases the realistic digital curves are not globally concave or convex. Probably, this approach can be applied for locally concave/convex smooth curves. This is subject for the future consideration.

3.3.3 Full search dynamic programming algorithm

For error measure L_2 the error of the approximation of segment $\{p_i, ..., p_j\}$ by the line segment (q_m, q_{m+1}) of Q is defined as the sum of squared Euclidean distances from each vertex of $\{p_i, ..., p_j\}$ to the corresponding line segment (q_m, q_{m+1}) (see Eq. 3.1.3). The approximation error $E_2(P)$ of the curve P by the curve Q is the sum of the errors of approximating each segment $\{p_i, ..., p_j\}$ of P by the corresponding line segment (q_m, q_{m+1}) of Q. The optimal approximation of curve P is then the set of vertices $\{q_2, ..., q_M\}$ of Q that minimizes the approximation error $E_2(P)$:

$$E_2(P) = \min_{\{q_m\}} \sum_{m=1}^{M} e^2(q_m, q_{m+1}) . \tag{3.3.1}$$

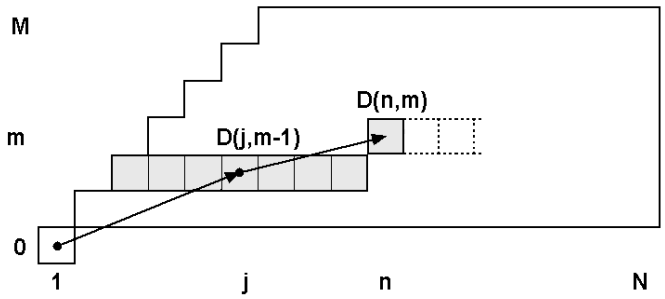

Figure 3.3.1: Scheme of computation of the cost function $D(n, m)$ in the state space Ω by dynamic programming algorithm of Perez and Vidal.

```
// Initialization
D(1,0) ←0
FOR  n = 2 TO  N DO
    D(n,0) ←∞
END
// Minimum search
```

```
FOR  m = 1 TO  M  DO
   FOR n = m TO  N  DO
      d_min ← ∞
      FOR  j= m–1 TO  n–1  DO
          d ← D(j, m–1) + e^2(i,j)
          IF(d < d_min)
             d_min ← d;
             j_min ← j
          ENDIF
       END
       D(n, m) ← d_min
       A(n, m) ← j_min
   END
END
// Backtracking for the solution H(m)
H(M)= N
FOR m = M  TO  1 DO
   H(m–1) = A(H(m), m))
END
E_2(P) ← D(N,M)
```

Figure 3.3.2: General scheme of full search dynamic programming algorithm of Perez and Vidal.

The optimization problem can be solved by the dynamic programming algorithm as proposed by Perez and Vidal [194] with the following recursive expressions (see Fig. 3.3.1):

$$D(n,m) = \min_{m-1\le j<n} \left\{ D(j,m-1) + e^2\left(p_j, p_n\right) \right\}; \quad m = 1,...,M; \tag{3.3.2}$$
$$A(n,m) = \arg\min_{m-1\le j<n} \left\{ D(j,m-1) + e^2\left(p_j, p_n\right) \right\}; \quad n = m,...,N.$$

Here $A(n, m)$ is the *parent state* that provides the minimum value for the cost function $D(n, m)$ at the state (n, m) of the state space Ω (see Fig. 3.3.1). Approximation vertices q_m and q_{m+1} of Q are vertices p_i and p_j of the correspondent line segment. The general scheme of the full search Perez and Vidal's algorithm is presented on Fig. 3.3.2.

3.3.4 Time and space complexity of Perez-Vidal algorithm

As it was mentioned above, time complexity of L_2-optimal Perez-Vidal algorithm is $O(MN^2)$. Let us consider problem of approximation in a trivial case with M=N-2. Perez and Vidal proposed the algorithm for approximation of digitized curves, where normally $M << N$, but generally speaking, in GIS applications (vector map

data reduction) any number of approximation segments can be given, including $M \approx N$.

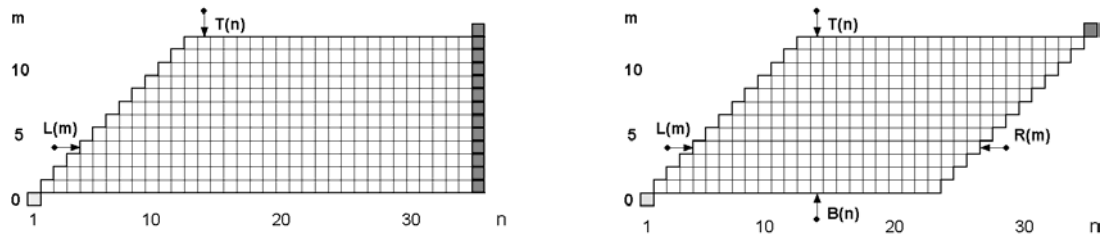

Figure 3.3.3: The original state space Ω in algorithm Perez and Vidal's algorithm (left); the modified *single-goal* state space Ω for *min-ε problem* in the proposed algorithm [**P4**] (right).

In fact, to obtain approximation with N-2 line segments we have to eliminate only one vertex from the input curve P (for example, with one step of Merge algorithm of Pikaz and Dinstein [198]). So, the best solution can be found in linear time by checking approximation error for every vertex as the eliminated one. On the other hand, complexity of the optimal algorithm for $M = N$-2 is given as $O((N\text{-}2)N^2))= O(N^3)$ time. The reason for the high complexity of the algorithm is the redundancy of the search: in the case under consideration we are constructing *all* solutions for m=1, 2, .., M-2, although in fact we need to know solution for single goal state Ω(N, M-2).

To solve the paradox, we proposed in [**P4**] a modification of the state space. The state space has to be bounded to eliminate states that are not necessary for the construction of the goal state (see Fig. 3.3.3, right). The complexity of the dynamic programming algorithm with the modified state space is $O(M(N\text{-}M)^2)$. As we can see, the time complexity of the modified DP algorithm for the trivial case in question is $O(N)$, as it should be (see Fig. 3.3.4).

In same cases, the input curve can be approximated with zero error by less number linear segments as a given M. In algorithm of Perez and Vidal such situation can be detected and the computation can be stopped even if the current number of segments m is less than M. In full search algorithm in modified state space algorithm, to be sure that a given number corresponds non-zero approximation error, we can check it with a simple procedure in $O(N)$ time by elimination of those points, whose absence does not affect on the total approximation error. If the found number $M_{\min}$ is bigger than a given number of linear segments M we can find approximation solution for this M, otherwise we can approximate the input curve by smaller number of segments $M_{\min}$ with zero approximation error.

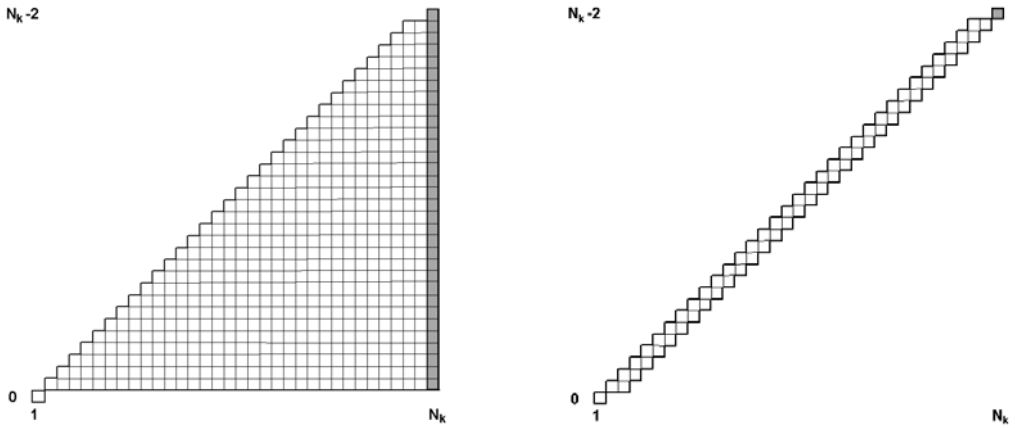

Figure 3.3.4: State space Ω for the case $N = M–2$: in Perez-Vidal algorithm (left), and in the proposed algorithm [**P4**] (right).

3.3.5 Using the preliminary computed error values

Haugland *et al*. [97, 98, 100] and Horng and Li in [111] offered to calculate approximation errors for all pairs of vertices in advance and store it in a 2-D array of $N \times N$ size. The reasons of this technique were different: Haugland *et al*. treat the *min*-ε *problem* as the search of the cardinality constrained shortest path in graph and all the weights in graph should be known prior the computations. The purpose of Horng and Li was to reduce processing time by performing all computations only one time and to use the stored values later.

The last idea seems to be reasonable, because it really permits to avoid recalculation of the values and reduce processing time. In practice, however, this method in the form suggested by Horng and Li works for relatively small N only because of high *space-complexity* of the method. Let us consider the following example of DP approximation: the 5004-vertex test shape #3 [231] is to be approximated by 50 linear segments. With algorithm of Perez and Vidal the approximation can be computed in 710 s [231]. According to the mentioned above approaches [97, 98, 111] the error values have to be stored in 2-D array of $N \times N$ size; it gives 200 Mb for the N=5000. Moreover, for approximation of closed contours Horng and Lee proposed to use 2-D array of $2N \times 2N$ size that means allocation of 800 Mb in the case under question. When RAM size is limiting resource, demands for allocation of 200 Mb (or even 800 Mb) can be a problematic way to reduce processing time.

Thus, for small N the precalculation method [97, 98, 111] does work, but only in the case when the processing time is small even without using the technique. For large N, however, the precalculation method cannot work efficiently because of high space complexity $O(N^2)$.

In [**P4**], we have proposed modification of DP algorithm of Perez and Vidal with precalculation by total cost of $O(MN)$ space instead of $O(N^2)$ as in [97, 98, 111]. Actually this space complexity is the same as that of the original algorithm, namely $O(MN)$.

Let us consider memory demands of the original Perez-Vidal algorithm in details. According to the scheme represented in the algorithm, the DP calculations are performed sequentially for all m starting from 1 to M. To support the calculation we need 2-D array of $M\times N$ size to store parent states $A(m, n)$ for the optimal sub-paths. For storing the cost function $D(m, n)$ it is enough to store 2-D table of $2\times N$ size. This is because we need to know only the previous row to calculate the current one. We also need 2-D table of $5\times N$ size for five arrays of cumulatives of coordinates to calculate approximation error *on-fly*. The total space complexity of Perez-Vidal algorithm is $O(MN)$.

According to the approach of Horng and Li [111] with storing of precalculated errors we need an additional 2D array of $N \times N$ size for the errors, which increases the total space complexity to $O(N^2)$. In addition, in [97, 98, 111] for the cost function $D(m, n)$ instead of $2\times N$ array was used 2-D array of $M\times N$ size.

Table 3.3.1. Space demands for DP algorithms for *min-ε problem* with precalculation.

	Full search [1)]	Full search [**P4**]	Reduced search [**P4**]
Cost function D	$O(MN)$	$O(MN)$	$O(WN)$
Parent states A	$O(MN)$	$O(MN)$	$O(WN)$
Cumulatives	$O(N)$	$O(N)$	$O(N)$
Errors $e^2(p_i,p_j)$	$O(\boldsymbol{N^2})$	$O(N)$	$O(N)$
Total space	$O(\boldsymbol{N^2})$	$O(MN)$	$O(WN)$

[1)] [97, 98, 111]

Now let us change the order of processing from *row-by-row* to *column-by-column*. We will fill the state space with solutions of sub-problem *column-by-column*, at first for n=2, then for n=3, until we reach the last vertex n=N. For every current vertex n, we have to find solutions of the sub-problems for all allowable values of m in the state space. For this computational scheme we need 2-D array for $A(m, n)$ of size $M\times N$ and five arrays of cumulatives of coordinates as in the original algorithm. Now we need 2-D array of size $M\times N$ for the cost function $D(m, n)$, not $2\times N$ as in the previous case. For the processing of the current vertex, we need

approximation errors for linear segments from the *current vertex* to the vertices already passed. We do not need anymore the approximation errors from all vertices to all ones for processing the current state. For this purpose it is enough 1-D array of size $1 \times N$. To support the storing pre-calculated cost function values, we have to use $O(MN)$ memory for the cost function $D(m, n)$.

Comparing space demands for two DP algorithms with full search (see Table 3.3.1), we can use precalculation method at the cost of $O(MN)$ additional memory instead of $O(N^2)$ as suggested in [97, 98, 111]. For the considered above example of approximation of 5000-vertex curve by M=50 segments with the proposed scheme in [**P4**] we need additionally about 1 Mb, that is only 1% of amount we must allocate with the approach of Horng and Li.

3.3.6 Summary

Perez and Vidal have proposed optimal DP-based algorithm which can solve the *min*-ε *problem* with L_2 error metrics in $O(MN^2)$ time. We introduced two improvements to the core algorithm, including modification of the state space to reduce complexity of the algorithm to $O(M(N-M)^2)$, and a scheme for the reducing processing time by storing of pre-calculated approximation errors. In the case of *min*-ε approximation of locally concave/convex smooth curves, it is worth further studies, if we can apply the matrix search algorithm to reduce processing time.

3.4 Iterative reduced search

3.4.1 Optimality versus time performance

Heuristic algorithms are fast, but they cannot provide optimal solution. Optimal algorithms of complexity $O(N^2)$–$O(N^3)$ are too slow to be used for large input. As it was already noticed by Heckbert and Garland who wrote [101]: "*Optimal simplification typically has quadratic or cubic cost, making it impractical for large inputs*". Zhang and Guo [312] also wrote that, in practice, the vertices number of a curve which DP or other exact optimal methods can tackle is about a 100 points. They have not provided information concerning processing time for test shapes they used. Nevertheless, the note is symptomatic: time complexity of optimal algorithms is high. Understanding that fact, the authors of optimal methods have also proposed a number of approaches to reduce processing time at the cost of optimality.

Perez and Vidal [194] in their concluding remarks have mentioned two possible techniques for obtaining *controlled* reduction in computational cost. The one is beam search, which essentially consists of discarding those branches that lead to an error greater than a certain margin at any given stage. The other method is to apply adjustment window that limits the maximum and minimum number of points assigned to any given edge.

Haugland *et al*. [97] motivates the development of optimal algorithms that the optimal algorithms can serve as a powerful tool when analyzing possible heuristic algorithms. He proposes the following approaches for further studies: a) divide the samples into K parts and perform processing on each of these; the execution time is reduced by about K^2; b) start with the path involving the first and last vertices only, and augment it gradually by one vertex until an M-vertex path is achieved; c) start with an arbitrary path with M segments and change the vertices on the path successively so that the path length is gradually reduced. Haugland *et al*. developed algorithm for approximation of ECG signals. In the case of quasi-periodical ECG signals, this approach is quite satisfactory, but in the case of arbitrary planar (or space) curve it can cause significant lost of quality.

Mori *et al*. [171] presented algorithm for optimal approximation of curves by linear segments, arcs and splines proposed 5:1 decimation of curve vertices to

reduce processing time. Schroeder and Laurent [243] proposed following two-step scheme for *min-# problem*: reduce the number of vertices using the approximation algorithm with bigger value of error tolerance ρε, where ρ>1, and apply the approximation algorithm to the obtained curve with a given error tolerance ε.

Salotti [230, 231] used preliminary approximation of the input to get upper limit for the approximation error to be used further in the process of the state space exploration. He also introduced two heuristic functions for cost function estimation to reduce the search. His A*-search algorithm is optimal but the complexity is still $O(N^2)$ even in the best case.

3.4.2 Paradigm of bounding corridor

Optimal algorithms are slow, whereas heuristic algorithms are fast but they lack the optimality. There have been attempts to improve the quality of heuristic algorithms by using local search or stochastic optimization techniques (see vertex-adjustment methods), but these methods cannot guarantee optimality or high fidelity. On the other hand, there have been attempts to reduce the processing time of DP algorithms by cost of optimality.

Polygonal approximation which is very close to the optimal one, is quite enough in most practical applications, because the real accuracy of output polygonal approximation is defined by the following factors: a) fidelity of the approximation algorithm, and b) accuracy of input vertex data (digitized curves, vector map). The quest for 100% fidelity is justified from mathematical point of view, but this demand can be released in practical applications if we take into consideration all technical details of the application and the time cost of the optimal result.

In **P4**, we try to bridge the gap between the optimal and heuristic algorithms by introducing a new paradigm of *bounding corridor* in the state space. Instead of time-consuming search in the full state space (see Fig. 3.4.1), we offer to perform the search only in the most relevant part of it, bounded by a corridor (see Fig. 3.4.2).

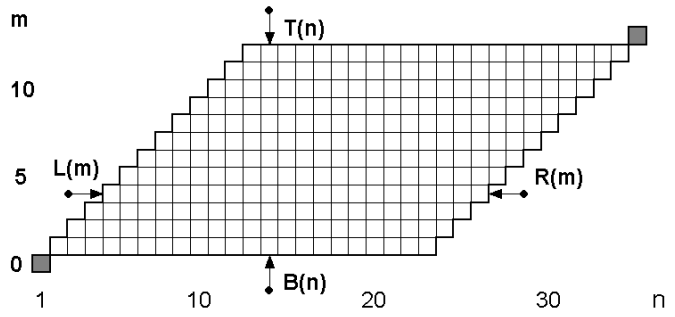

Figure 3.4.1: Illustration of the modified single-goal state space Ω for a sample problem size of N=35, M=13. The start and goal states are marked with the gray squares.

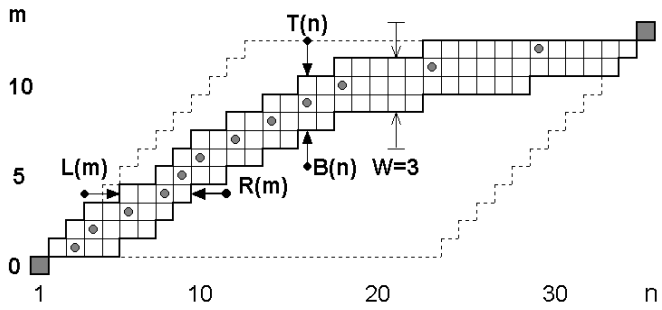

Figure 3.4.2: Illustration of the bounding corridor of width W=3 in the state space Ω. The reference path H is marked by the gray circles; the single-goal state space Ω is marked by the dashed line.

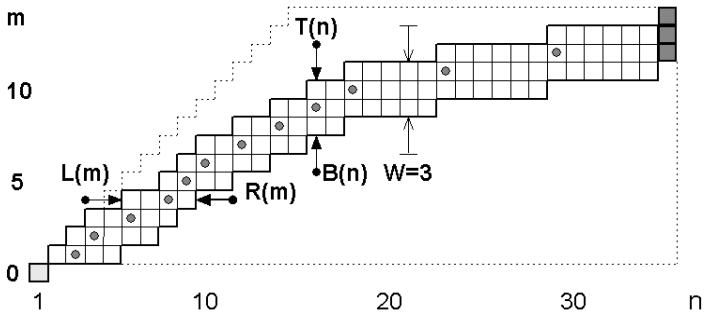

Figure 3.4.3: Illustration of the *multiple-goal* bounding corridor of width W=3 in the state space Ω. The reference path H is marked by the gray circles; the multiple-goal state space Ω is marked by the dashed line.

The corridor of fixed width W in the state space is constructed along a *reference path*, which can be obtained by any fast heuristic algorithm.

In contrast to constraints on approximation error [230, 231], we use geometrical constraints on search area in the state space to control the breadth of the search. Width of the corridor W at some approximation point defines the range of possible numbers of approximation segments for the point. At the same time, location of the approximation points is optimized too. In the mentioned algorithms with local adjustment of approximation points the number of segments assigned to curve part from the first vertex to approximation point is the same, only the location of the approximation point can be changed within a narrow range.

The optimization algorithms [44, 153, 173, 110] can be formally considered as a special case of the search in narrow non-continuous corridor. On the other hand, the reduced search algorithm with reference approximation can be treated as optimization of the preliminary (reference) solution. The main difference of the reduced search algorithm from the vertex adjustment method is not only *quantitative* (wider range of the search), but it is *qualitative* one too. The introduced paradigm of bounding corridor allows us to solve a number of approximation problems, which cannot be solved by mentioned above vertex adjustment and search reduction methods because of the following reasons.

a) The proposed reduced search algorithm can be iterated using the output solution as a new reference path in the next iteration. The number of iterations can be given in advance, or adaptively varied depending on the development of the approximation error. With the iterative reduce search we can achieve practically optimal solution in $O(N) \div O(N^2)$ time [**P4**]. With vertex adjustment method we cannot improve obtained solution by additional iterations.

b) The presented *single-goal* bounding corridor can be extended to the case of *multiple-goal* corridor (see Fig. 3.4.3). With the search in the multiple-goal bounding corridor a family of solutions can be obtained, that is defined by the corridor width W. This kind of corridor have been further used for solving the multiple object *min*-ε *problem* [**P7**].

c) In the case of *min*-ε *problem* for closed contours, the corresponding analysis of solutions in the bounding corridor can be performed to find the sub-path with conjugate states that provides minimum of cost function (approximation error). This sub-path gives optimal approximation solution to the closed contour, including optimal selection of the starting point [**P6**].

Approximation of locally concave/convex curves

In the section 3.3.2 it was mentioned that matrix search algorithm might be applied to reduce processing time for approximation of locally concave/convex curves. This

idea fits perfectly to the reduced search approach. Performing search in the bounding corridor we involve some part of the curve into the computation. If this part is concave (or convex) the search for the minimum, could be speed-up with matrix search algorithm [7, 8, 297, 298]. Information obtained at the first run of reduced search about the location of concave/convex parts could be used in subsequent iterations. Joint using of matrix search algorithm and iterative reduced search approach for locally concave/convex curves to reduce processing time is a topic for future studies.

3.3 .3 Summary

Aiming to bridge the gap between slow optimal and fast heuristic algorithms for *min*-ε *problem* we introduced paradigm of bounding corridor and iterative reduced search approach. We can control trade-off between optimality and time performance with parameters of the proposed algorithm. The time complexity of the algorithm is between $O(N)$ and $O(N^2)$, that corresponds the complexity of fast heuristic algorithms. The processing rate with the proposed algorithm can be roughly estimated as 10^3–10^4 vertices per second depending on relative number of segments, solution fidelity, curve smoothness, and processor performance. On the other hand, with the proposed algorithm we can achieve optimal or near-optimal result, using a proper strategy.

Later we will use the iterative reduced search for the development of efficient algorithms in polygonal approximation field, namely, *min*-# *problem* for open curves, *min*-ε and *min*-# approximation of closed contours, and *min*-ε approximation of multiple objects.

3.5 Optimal algorithms for *min-# problem*

The *min-# problem* is formulated as follows: a given polygonal curve P, approximate it by another polygonal curve Q with the minimum number of segments M so that the approximation error does not exceed a given maximum tolerance ε. The *min-# problem* is motivated by necessity to obtain approximation with least number of segments that maintans a certain level of accuracy. The problem arises in practical task of vectorization, vector data reduction, and vector map simplification.

3.5.1 Survey of solutions

Probably, one of the the first optimal algorithms for *min-# problem* for error criterion L_∞ has been proposed by Papakonstantinou in 1985 [187]. The *min-#* problem under uniform L_∞ measure (in fact, with infinite beam criterion) was formulated as optimization task and was solved by dynamic programming (DP) method. To reduce DP search, the longest line segment is defined by cone-intersection method of Sklansky and Gonzalez [250].

One year later Dunham [73] published *min-#* algorithm that uses DP approach for optimal approximation with least number of segments for error measure L_∞. He used modified scan-along algorithm of [290] and [250] with cone intersection to reduce DP search. The complexity of the algorithm is $O(N^3)$ in the worst case.

About the same time Imai and Iri [117-119] present a unified approach for the problem by formulating it in terms of *graph theory*. At first, a directed acyclic graph $G_\varepsilon(P) = (V, E_\varepsilon)$ is constructed (see Fig. 3.5.1, left). Nodes V are vertices of the curve $P = \{p_1, \ldots, p_N\}$ and edges E_ε are the approximation line segments $E_\varepsilon = \{(p_i, p_j): 1 \le i < j \le N \mid D(p_i, p_j) \le \varepsilon\}$. Two nodes p_i and p_j of $G_\varepsilon(P)$ are connected with an edge (p_i, p_j), iff the correspondent approximation error (maximum deviation) $D(p_i, p_j)$ is at most the prescribed error tolerance ε: $D(p_i, p_j) \le \varepsilon$.

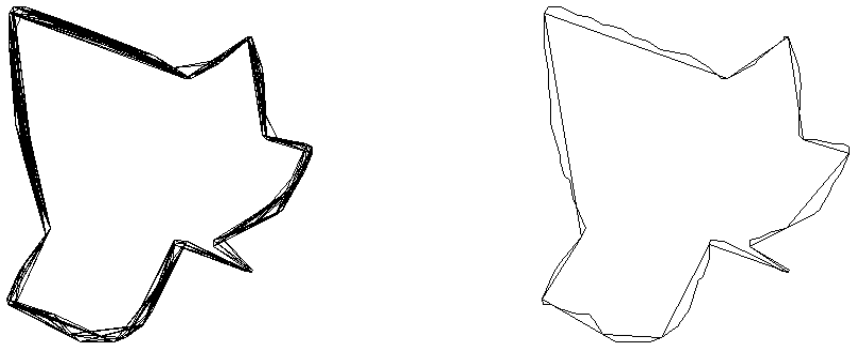

Figure 3.5.1: Graph $G_\varepsilon(P)$ constructed on digitized curve P for error tolerance ε=10 (left); *min*-# approximation solution for ε=10 as the shortest path in the graph $G_\varepsilon(P)$ (right).

Solving *min*-# problem consists of finding the *shortest path* from p_1 to p_N in digraph $G_\varepsilon(P)$ (see Fig. 3.5.1, right). Since the graph is acyclic, this path can be found in time proportional to the number of edges [64, 55]. The brute-force method of constructing $G_\varepsilon(P)$ is to check for each pair of vertices p_i and p_j whether the error $D(p_i, p_j)$ is within error tolerance ε. There are $O(N^2)$ pairs of vertices and checking the error, corresponding to a pair, takes $O(N)$ time. This brute-force method for the graph $G_\varepsilon(P)$ construction takes $O(N^3)$, while finding the shortest path in G takes no more than $O(N^2)$ time. Thus, the bottlenck in the computation of *min*-# approximation is the construction of the graph $G_\varepsilon(P)$. The main efforts of researchers have concentrated on the problem of how to reduce complexity of the graph construction.

Melkman and O'Rourke [165] studied the 2-D *min*-# and *min*-ε problems. Their version of Imai-Iri's algorithms takes $O(N^2 \log N)$ for *min*-# *problem*, and $O(N^2 \log^2 N)$ for *min*-ε *problem*. Space complexity is $O(N^2)$ in both cases. Thus, the graph construction was reduced from $O(N^3)$ to $O(N^2 \log N)$. They proposed cone-intersection algorithm to reduce the complexity of the graph constuction. This algorithm is analog of that of Sklansky and Gonzalez [250].

Chan and Chin [37] reduced the time complexity to $O(N^2)$ for *min*-# *problem*, and $O(N^2 \log N)$ for *min*-ε *problem* using the parallel-strip criterion They improved complexity of constructing graph G to $O(N^2)$. They have also shown that for closed polygonal curve the *min*-# *problem* can be solved in $S(N)$ time, where $S(N)$ denotes the time complexity for solving all-pairs shortest path problem of $G_\varepsilon(P)$. They also presented algorithm of linear time complexity for *min*-# problem for the convex curves (see Table 3.5.1).

Table 3.5.1: Summary of the time complexities of the polygonal algorithms of Chan and Chin [37].

Shape type	General		Convex	
	Open	Closed	Open	Closed
Min-# problem	$O(N^2)$	S(N)	$O(N)$	$O(N^2)$
Min-ε problem	$O(N^2 \log N)$	S(N)log$N$	$O(N^2)$	$O(N^2)$

Based on the *infinite beam criterion*, Toussaint [269] solved the *min-# problem* in $O(N^2 \log N)$ time and $O(N^2)$ space. Imai and Iri gave an $O(N^2 \log^2 N)$ time and $O(N^2)$ space for the *min-ε problem*. Eu and Toussaint [77] published another algorithm for *min-# problem* under the infinite beam criterion that was claimed to take $O(N^2)$ time, and $O(N^2 \log N)$) for *min-ε* problem. The space complexity is $O(N^2)$. Eu and Toussaint [77] also used the infinte beam criterion based on L_1 and L_∞ distance metrics. Using L_2 distance metric, the *min-# problem* can be solved in $O(N^3)$ time, and *min-ε* problem in O($N^3 \log N$)) time.

Ray and Ray [212] determined least number of segments with the minimum possible error by maximizing an objective function comprised of the length of line segments and the sum of absolute errors between the line segments and the digital curve (L_1 error measure).

Pikaz and Dinstein [197] found *min-#* approximation of the polygonal curve where the city-block metric is used to measure distance between the approximation and the input curve. For the city-block the distance the complexity of the algorithm is $O(N^2)$.

Chen and Daescu [40, 41] presented a number of efficient algorithms for the 2-D *min-#* and *min-ε problems* in the same time bounds as [37, 77], but using only $O(N)$ space in comparison with $O(N^2)$. Also they applied the technique to a special case of the 3-D *min-#* and *min-ε problems*.

Zhu and Seneviratne [320] have shown that cone-intersection algorithm [250] has the problem that some peaks will disappear from the curve when the cone angle is the only evaluation factor and introduced modification to Sklansky and Gonzalez method. Based on the modified algorithm for they proposed new version of the algorithm for *min-# problem*.

Katsaggelos *et al*. [132] used polygonal approximation for lossy encoding with minimum rate and given distortion bounds. Actually, Katsaggelos *et al*. solved more general problem, than just a *min*-# one. In this case, the techniques based on geometrical computations, cannot be used. Because of the brute-force method applied for construction of the graph $G_\varepsilon(P)$, time complexity of their algorithm is $O(N^3)$.

Hosur and Ma [114] following the approach of Katsaggelos *et al*. for *min*-# problem proposed cone intersection method for reducing the graph construction time, which actually has been introduced early in [250] and have already been used in other algorithms for *min*-# *problem* [187, 73].

Recently Agarwal *et al*. [6] proposed *min*-# algorithm for Hausdorff and Frechet error measure of $O(N^{4/3+\delta})$ time complexity, where $\delta>0$. Barequet *et al*. [22] presented agorithms for approximating polygonal curves in 3D and higher dimensional spaces under the tolerance zone criterion.

Table 3.5.2: Summary of *min*-# and *min*-ε results from [22].

Metric	3 dimensions		$d\geq4$ dimensions	
	min-#	*min*-ε	*min*-#	*min*-ε
L_2	$O(N^2\log N)$	$O(N^2\log^3 N)$	$O(N^{3-2/(\lfloor d/2\rfloor+1)}\text{polylog}N)$	$O(N^{73}\text{polylog}N)^*$
L_1 & L_∞	$O(N^2)$	$O(N^2\log N)$	$O(N^2)$	$O(N^2\log N)$

*For d=4 only.

With DP algorithm of Perez and Vidal the *min*-# *problem* for measure L_2 can be solved in $O(N^3)$ time (see also [274]). Salloti proposed fast optimal algorithm for the problem based on the A*-search algorithm, and he introduced for *min*-ε *problem* with L_2 measure [230, 231]. Using the heuristic error estimations to stop the search, he reduced complexity of Perez-Vidal algorithm from $O(N^3)$ to $O(N^2)$ time.

3.5.2 Problem of multiple solutions

As we can see, the error measure L_∞ is widely used in heuristic and optimal algorithms for *min*-# *problem*. The error measure L_∞ is preferred to other measures for obvious reasons. But careful study of these *min*-# algorithms based on the error measure L_∞ shows that all the algorithms suffer from an essential drawback, namely visible distortion of approximation curve (see Fig. 3.5.2, left)

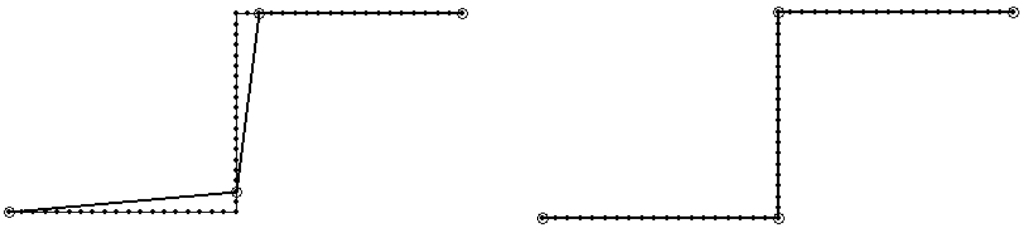

Figure 3.5.2: Examples of *min*-# approximation of 59-vertex test shape for error tolerance ε=2 with L_∞ error metrics: by algorithm based on the shortest path in digraph (left); by the proposed method [**P5**] with L_2 optimization (right). Vertices of the input curve are labeled with dots; the approximation points are labeled with circles.

Firstly, the drawback was noted by Papakonstantinou [188] who has been studying problem of data reduction of ECG signals by polygonal approximation with minimum number of line segments for measure L_∞. As it was mentioned above, he wrote, that so called optimal solution is not unique. Actually, the number of the equivalent optimal solutions (having the same minimal number of line segments) can be extremely large (see Fig. 3.5.3). The reason of the effect is as follows: with the error measure L_∞ we can control only the maximum deviation, but not the deviations for all the vertices of the approximated curve. Formally, the solutions satisfy the constraint on maximum deviation, but the solutions have visible shape distortion.

To overcome the problem, Papakonstantinou proposed how to select the best one among the formally optimal solutions, with the minimal integral square error. In other word, he proposed to use error measures L_∞ and L_2 jointly to obtain solution, which satisfies the condition on maximum deviation $D(P) \le \varepsilon$ and provides minimum to the approximation error $E(P)$ with measure L_2. The complexity of the modified *min*-# algorithm is defined by the complexity of the base algorithm but the processing time is bigger because of L_2-optimization. The proposed method provides good results. It is easy to understand why the effect has been discovered by studying approximation of ECG signals. For smooth curves the distortion effect is small, whereas for curves with sharp corners, such as ECG signals, the effect is clearly visible.

Figure 3.5.3: The test 2900-vertex shape ”Australia”: all shortest paths in the graph $G_\varepsilon(P)$ for a given error tolerance ε=1.

Haugland *et al*. [97, 179] attacked the problem by joint use of measures L_∞ and L_2 from another side. They solved problem of ECG data reduction with a given data compression ratio by algorithm for *min*-ε *problem* with measure L_2. Proximity of the approximation curve to the input one is measured by the sum of squared errors, but the deviation (error value) for every individual vertex is beyond the control of the algorithm. To overcome the problem, they proposed modified version of the base DP algorithm for *min*-ε *problem*, introducing additional check if the local deviation less than the tolerance level. To reduce the processing time for the computation of the L_∞ approximation error, a fast algorithm based on convex hull construction has been used.

Authors formulated a new version of *min*-ε *problem* using two conditions: a) the number of segments should be less than a given value: $M \le M_0$, and b) the maximum deviation should be below the tolerance level: $D(P) \le \varepsilon$. As we can see, the conditions are contradictious: if the given number of segments M_0 is too small, the constraint on maximum deviation cannot be satisfied with any number of segments. The problem can be solved with constraint on the maximum error, or on the number of segments, but not on both of them.

Pikaz and Dinstein in [197] solved *min*-# *problem* as the shortest path on digraph with city-block distance metrics. The obtained polygonal approximation is minimal with respect to the number of vertices under a given maximal error. As in the mentioned above case, there might be several different polygonal approximations with the same minimal number of vertices. Pikaz and Dinstein offered to select solution that is optimal according to the criterion of *minimal maximal* distance

between the approximation and the original curve. Although the proposed method allows reducing the maximum deviation, the local deviations (smaller than tolerance level) are still out of control.

3.5.3 Selection of the best solution with reduced search

In [**P5**] we apply the iterative reduced search for solving the *min-# problem*. At first, the initial solution approximation for a given error tolerance ε is obtained with any algorithm for *min-# problem*, then a bounding corridor is constructed along the reference path, and finally the solution is searched in the bounding corridor (see Figs. 3.5.2, right, and 3.5.4, right).

Experiments have shown, that although the integral squared error $E(P)$ is reduced after L_2-optimization, the maximum deviation $D(P)$ can be bigger than a given tolerance level in a few outliers. The problem can be solved by additional iterations with reduced search to increase the number of linear segments M. Another solution is to perform L_2-optimization with L_∞ constraints on the approximation error by cost of higher time complexity because of L_∞ approximation error calculation.

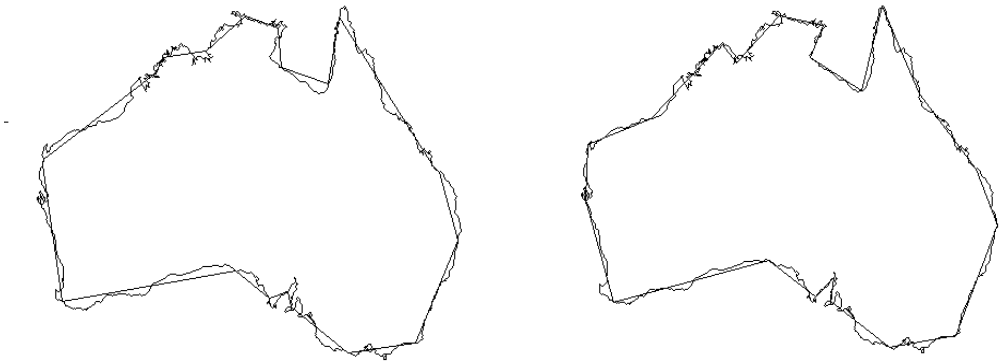

Figure 3.5.4: The *min-#* approximation for ε = 10: the solution as one of the shortest path in graph $G_\varepsilon(P)$: processing time: $D_1(P)$=1, E_1=629, T_1=4.5 s (left); this solution after processing with reduced search DP algorithm: E_2=298, maximum distance $D_2(P)$=1.08; additional processing time T_2=0.5 s (right). The number of vertices N = 2900.

In high quality vectorization tasks, the increase of the maximum deviation is about 10% and takes place for outliers only. In our opinion, in practical applications, the given constraint of maximum deviation ε can be treated in non-strong way. In other words, to reduce processing time we can use the iterative reduced search to optimize location of vertices. The obtained approximation points satisfies the given constraint on the deviation, excluding may be a few outliers.

3.5.4 Summary

We have proposed to use iterative reduced search approach to improve the quality of *min*-# approximation solutions obtained. The initial min-# solution can be found with any algorithm with L_∞ error measure. Then we optimize location of the approximation vertices with L_2 error measure using iterative reduced search. The proposed algorithm is tailored for high-quality vectorization of digitized curves.

3.6 Approximation of closed contours

3.6.1 Problem formulation

A closed *N-vertex polygonal curve P* in 2-dimensional space is represented as the ordered set of vertices $P=\{p_1, \ldots, p_N \mid p_N \equiv p_1\}$, when the last vertex is equal to the first one. The approximation curve Q consists of $(M+1)$ vertices: $Q=\{q_1, \ldots, q_{M+1} \mid q_{M+1} \equiv q_1\}$, where the set of vertices q_m is a subset of P. In the case of *closed* contours, we have to find optimal allocation of all approximation vertices including the starting point (see Fig. 3.6.1).

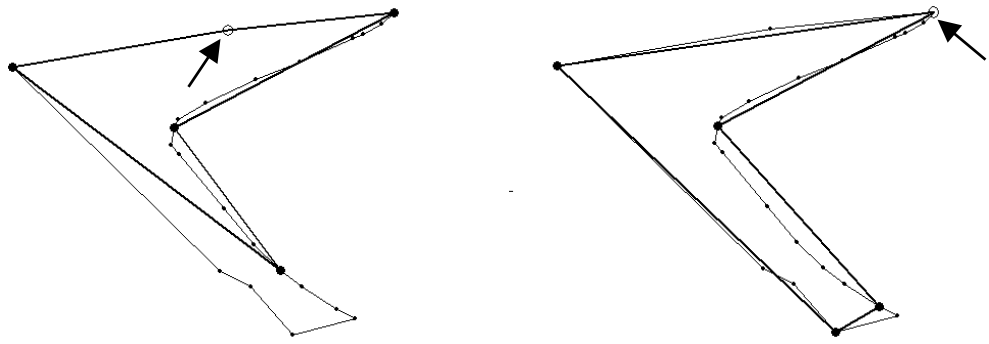

Figure 3.6.1. Examples of *min*-ε approximation of the closed contour #1 by $M=5$ linear segments: with non-optimal starting point (left); with the optimal starting point (right). The starting points are marked with arrow.

3.6.2 Survey of solutions

Imai and Iri [119] wrote that the closed curve *min*-# (resp. *min*-ε) problem can be solved by solving the N ordinary open curve *min*-# (resp. *min*-ε) problems. If complexity of algorithm for open curve *min*-# *problem* is $O(N^2)$, the complexity for closed contour approximation is $O(N^3)$ time. Perez and Vidal [194] also wrote that the complexity of the straightforward *min*-ε algorithm for closed contour is $(N-M)$ times that of the algorithm for open curve. Taking into account the complexity of modified full-search DP algorithm [194, **P4**] for closed contours *min*-ε problem it gives time complexity of $O(M(N-M)^3)$.

It was shown in [37] that the *min-# problem* for closed curve can be solved in $S(N)$ time, where $S(N)$ is the time for solving the all-pairs shortest path problem for a graph of N vertices.

There also exist a number of heuristic approaches for selecting the starting point. Sato [238] chooses the farthest point from the center of gravity as a starting point. Ray and Ray [212] in their formulation of the approximation problem specify neither the error nor number of line segments as cost function, but a ratio of the local L_1 error measure to the length of the segments. For the case of the closed contours they proposed to extend the sequential search beyond the current starting point until to the first approximation vertex to revise the choice of the starting point.

Pikaz and Dinstein [197] considered *min-# problem* with city-block error metrics, and proposed an algorithm based on the shortest path problem, including the search of optimal starting point by an algorithm of complexity $O(N^2)$. Zhu and Seneviratne [320] considered the approximation problem with L_∞ metrics, and proposed an $O(N^2)$ time algorithm for open curves. An iterative procedure for optimizing the choice of the starting point was also suggested for the case of closed curve. Schroeder and Laurent [243] in near-optimal algorithm for *min-# problem* perform preliminary approximation until the first approximation vertex is reached, and use this vertex as a new starting point for the second iteration.

Horng and Li [111] proposed a two-step method to select the starting point: at the first step, optimal approximation with any starting point is performed. At the second step, the approximation is performed again using the $\lfloor M/2 \rfloor$-th vertex as the new starting point.

To sum up, the proposed heuristic approaches for closed curves are sub-optimal whereas the optimal choice of the starting point is time consuming. Thus, the problem has not been solved satisfactory. One of the heuristic approaches is to perform preliminary approximation using one of the approximation vertices as a new starting point for final approximation [212, 197, 320, 111]. With this approach for starting point selection it is possible to obtain good results but, in general, the optimality of solution cannot be guaranteed (see Fig. 3.6.2).

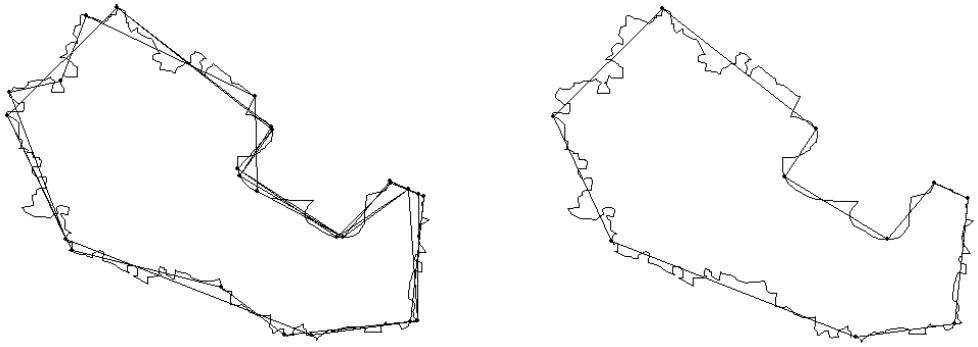

Figure 3.6.2: Results of *min*-ε approximation of 417-vertex closed contour # 2 by M=10 linear segments for all the possible initial starting points: by heuristic method [111] (left); by the proposed method for optimal staring point allocation [**P6**] (right).

3.6.3 Analysis of state space for *min*-ε *problem*

In the mentioned heuristic algorithms after the first iteration, however, the search starts from scratch and loses the information of the previous run. The idea of the approach we proposed in [**P6**] is to extend dynamic programming search in special state space beyond the last vertex of contour in cyclic way until the last point will be reached again. Solutions for sub-tasks in the state space are analyzed to find the best possible starting point.

We call two states on the sub-path in state space *conjugate* if the sub-path with the states corresponds to approximation of the input contour with the same start and end points. Approximation error for the sub-path is defined as the difference of the cost function values of the conjugate states. To find the optimal starting point we have to find two conjugate states, which provide minimum of the approximation error $E(m)$ for sub-path defined for all goal states in the state space (see Fig. 3.6.3). As we know, in DP algorithm for *open* curve solutions are constructed under the condition $q_{M+1}=p_N$. Propagating DP search in state space cyclically beyond the end point of P, we remove the restriction and make the approximation vertex q_{M+1} "free". Then we analyze solutions of all relevant sub-tasks with the free vertex q_{M+1} to find the best location for the starting point. If the original starting point gives the optimal solution, the correspondent path to goal state (N,M) is limited by conjugate states.

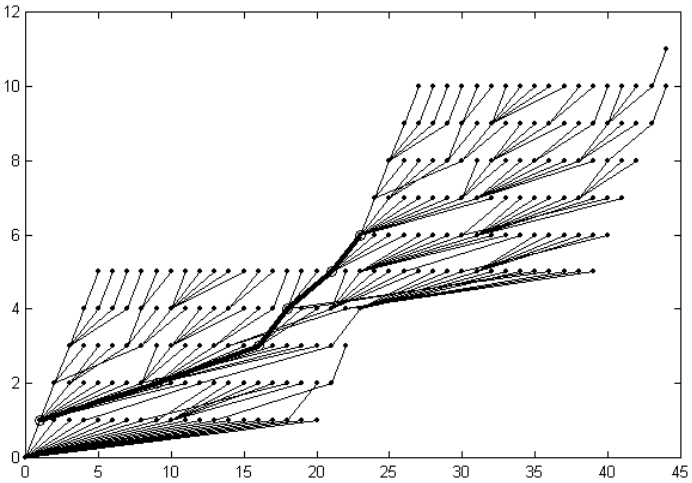

Figure 3.6.3: State space Ω_2 for the closed contour #1. Optimal sub-path in the space Ω_2 for the conjugate states with minimum cost function is emphasized with thick line and circles. Approximation with the optimal starting point that corresponds the sub-path is on Fig. 3.6.1, right.

Generally speaking, global optimality of the found starting point cannot be guaranteed. Moreover, the sub-path with conjugate states may be does not exist in the state space Ω_2. Sometimes it happens in the case of coarse approximation when the number of vertices is big ($N \sim 1000$) and the number of vertices is relatively small ($M < 10$). We can extend DP search cyclically to the next runs along the curve in attempt to find better solution in the state space Ω_k, where $k \geq 3$. Anyway, the state space contains solution for the original starting point which can be use.

3.6.4 *Min*-# approximation of closed contours

To solve the *min-# problem* for closed curves, we have to find approximation of the closed contour P by another closed contour Q with minimal number of line segments with an approximation error within given error tolerance level: $D(P) \leq \varepsilon$. The approximation error $D(P)$ with measure L_∞ is given as the maximum Euclidean distance from the vertices of P to the approximation linear segments.

To solve the *min-# problem* a digraph is constructed on the vertices of the input contour P. In the digraph, a pair of vertices p_i and p_j are connected with an edge if the approximation error of the curve segment between the vertices is less than a given error tolerance: $d(p_i,p_j) \leq \varepsilon$. The optimal solution is then a given by solving the shortest path in the digraph. This can be solved by using DP algorithm for the shortest path problem in the digraph.

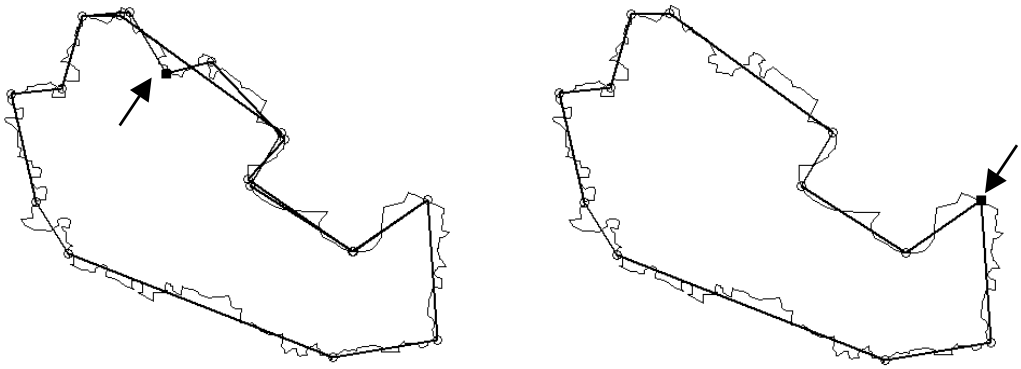

Figure 3.6.4. Result of *min*-# approximation of double-size curve P_2 for error tolerance ε=25 (left). *Min*-# approximation of the input closed contour #2 with optimal starting point obtained by analysis of DP solutions for P_2 (right). Starting points are marked with arrow.

To find the optimal approximation for closed contour we shall follow the approach introduced in [**P6**]: perform approximation of the wrapped double-size closed contour P_2 and then analyze the state space. In the case of min-# problem, the analysis of the space is reduced to the analysis of the solutions for the relevant sub-problems: we have to found the part of the approximation polygonal curve Q_2, which have the same start and end points (see Fig. 3.6.4). As in the previous case, the initial approximation of the input curve can be extended cyclically to find better solution. Even if such a solution with less number of segments is not found, the obtained solution for the original starting point anyway can be used.

3.6.5 Summary

We have introduced a new approach for *min*-ε and *min*-# approximation of closed contours based on dynamic programming method for open curves. It performs approximation of the cyclically extended double-size contour and then makes analysis of the state space to select the best starting point. The processing time is double of that of the approximation of the corresponding open curve. The time complexity of the algorithms is defined by the complexity of approximation algorithms for open curves in use. For solving the *min*-ε *problem* the suggested method can be used along with iterative reduced search algorithm with time complexity between $O(N)$ and $O(N^2)$.

3.7 Multiple-object polygonal approximation

3.7.1 Problem formulation

The problem of polygonal approximation of a single curve can be extended to the case of multiple curves (see Fig. 3.7.1) as follows:

> ***Multiple object min-# problem***: Given K polygonal curves $P_1, P_2, ..., P_K$, approximate it by set of K another polygonal curves $Q_1, Q_2, ..., Q_K$ with the minimum total number of segments M so that the approximation error does not exceed a given maximum tolerance ε.
>
> ***Multiple object min-ε problem***: Given K polygonal curves $P_1, P_2, ..., P_K$, approximate it by set of K another polygonal curves $Q_1, Q_2, ..., Q_K$ with a given total number of segments M so that the total approximation error is minimized.

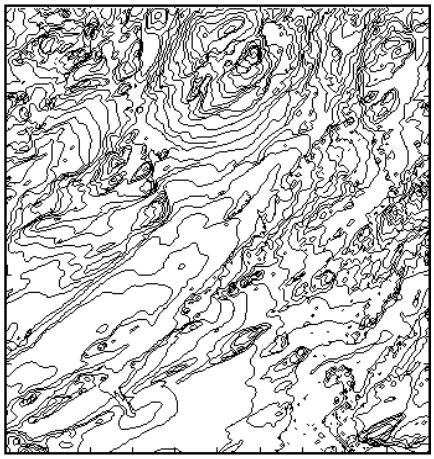

Figure 3.7.1: Example of multiple-object vector map: "Elevation map" [NLS], the total number of vertices N=38924, the number of objects K=569.

Solution for the *multiple-object min-# problem* depends on the error measure in use. In the case of L_∞ error measure, the problem reduces to the *single-object min-# problem* as the optimization can be solved for every object independently [244]. In

the case of additive error measures (L_1, L_2, *etc.*), on the other hand, the problem is not trivial. Fortunately, in practical applications we mostly have to deal with error measure L_∞ because the use of additive error measures (L_1 or L_2) is not practical in the case of *min-# problem*.

The case of *min*-ε approximation of *multiple objects* (with any error measure) is more complicated. The optimal approximation cannot be obtained by solving the approximation of each individual objects separately because the given total number of approximation segments should be optimally distributed among all objects. For example, uniform allocation of the segments can assign too many segments to the less complicated objects and, respectively, lacking the segments for more complicated objects. This situation is illustrated in Fig. 3.7.2.

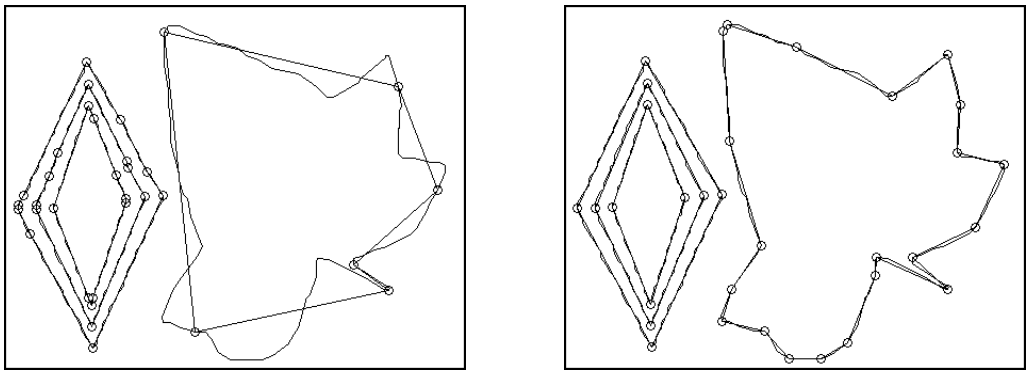

Figure 3.7.2: Example of multiple object approximation with uniform allocation of the segments numbers ($M_k \approx N_k M/N$); M_D=3×9 and M_L=6 (left), and with optimal allocation of the segments number (right), M_D=3×4 and M_L=21. The number of points in the objects are N_D = 3×121 ("Diamond"), and N_L = 82 ("Leaf").

The *multiple-object min-*ε *problem* can be formulated as the following optimization task for the total approximation error $E(P_1, \ldots, P_K, K)$ for K objects $\{P_1, \ldots, P_K\}$:

$$E(P_1,\ldots,P_K,M) = \min_{\{M_k\}} \min_{\{q_m\}} \sum_{k=1}^{K} \sum_{m=1}^{M_k-1} e^2\left(q_{k,m}, q_{k,m+1}\right) \text{ subject to } : \sum_{k=1}^{K} M_k \le M,$$

where $e^2(q_{k,m}, q_{k,m+1})$ is approximation error with measure L_2 of curve segment $\{p_i, \ldots, p_j\}$ of P_k by the correspondent line segment $(q_{k,m}, q_{k,m+1})$ of Q_k; M is a given total number of segments and M_k is the number of segments in object P_k.

3.7.2 Survey of solutions

Shuster and Katsaggelos considered problem of lossy encoding of object boundaries in rate-distortion sense [244]. The digitized contour is approximated by polygon Q, which leads to the smallest distortion for a given rate (number of bits). Actually, the case when the rate is proportional to the number of line segments corresponds to the *multiple-object min-ε problem*. Schuster and Katsaggelos [244] considered two different classes of distortion measures: L_∞ and L_2. For the first class (measure L_∞), they used scheme based on the shortest path algorithm for a weighted directed acyclic graph. For the second class (measure L_2) they proposed two algorithms.

The first algorithm is based on the Lagrangian multipliers method, which uses the DP algorithm for the shortest path in a directed acyclic graph. The time complexity of the Lagrangian approach for a fixed multiplier λ is the same as for the base shortest path algorithm. The shortest path algorithm is invoked several times by the bisection algorithm to find the optimal λ^* and, hence, the time complexity is a function of the number of required iterations. Thus, the complexity of the first algorithm is $O(N^2 \log N)$ because it is defined by the complexity of the shortest path algorithm and the number of bisection iterations. Since the Lagrangian multiplier method can only find solutions on the convex hull of the operational rate-distortion functions, they also proposed a tree-pruning based algorithm. This method is a one pass variant algorithm with the complexity of $O(N^2)$, but the efficiency of the pruning scheme cannot be guaranteed in general.

Algorithms with the complexity of higher than $O(N^2)$ can be used for relatively small input. This can be suitable for the encoding of object contours for MPEG-4 standard [132] but it can be too slow in the case of large vector maps.

3.7.3 Reduced search algorithm

In the **P7**, the *min-ε problem* of optimal approximation of multiple-object vector data was considered. We have introduced two algorithms for solving the problem based on dynamic programming: full search and iterative reduced search. Full search algorithm of complexity $O(N^3)$ has been introduced to represent the DP approach for joint optimization the number of segments and the approximation of the individual objects. At first, the rate-distortion functions are computed for all the objects as minimal approximation error for all possible number of segments. Then problem of optimal allocation of constrained resource is solved by DP algorithm. Finally the optimal approximation for every object is calculated for the found optimal distribution of segment numbers.

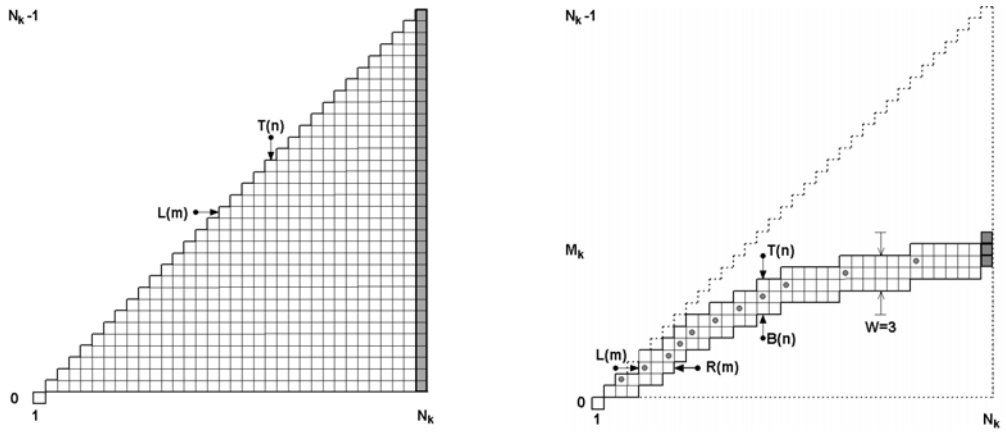

Figure 3.7.3: Illustration of the *multiple-goal* state space Ω_k for full search DP algorithm (sample problem of N_k=34 and M_k=12) (left), and the *multiple-goal* bounding corridor of width W=3 in the state space for reduced search DP algorithm (sample problem of N_k=34 and M_k=12) (right). The reference path $H(m)$ in the corridor is marked with dark gray circles, and the goal states with gray squares.

Because the time and space complexity of the full search algorithm is high, the iterative reduced search approach was generalized to the problem under consideration. We follow the main idea of the reduced search by reducing the search space by a given preliminary solution for the approximation, and then perform the search in the reduced space iteratively (see Fig. 3.7.3). Main difference to the full search is that a smaller search area is needed, which makes the algorithm faster. The iterative reduced search algorithm has time complexity of $O(N)$–$O(N^2)$. This is significantly smaller than the $O(N^3)$ of the full search, or the $O(N^2\log(N))$ of [244]. The reduced search approach is also applicable for very large data sets with reasonable memory requirements. Although the optimality of the algorithm cannot be guaranteed in general, the experiments indicate that the method is capable of finding the optimal solution even in the case of very large data sets (see Fig. 3.7.4). The algorithm can also be tuned for obtaining very fast sub-optimal solutions by reducing the number of iterations and corridor width.

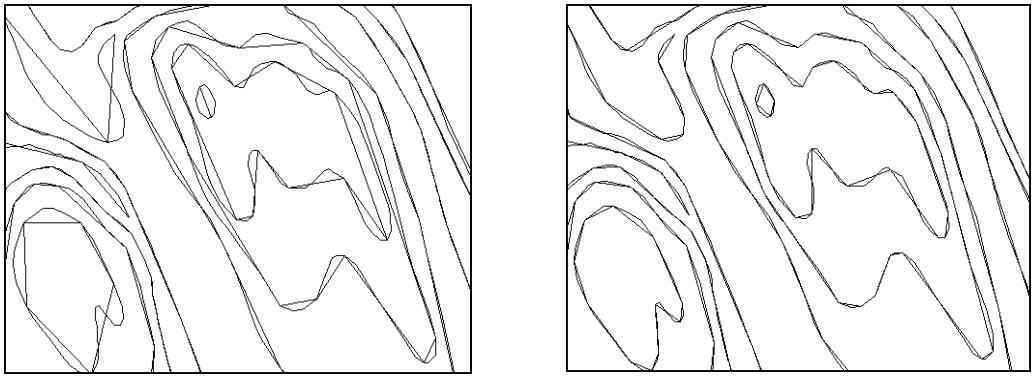

Figure 3.7.4: Approximation results for test data “Elevation map”: solution (fragment) for straightforward approach, Douglas-Peucker approximation with uniform allocation of the segments numbers: $E_0 = 892158$, $T_0 = 1.7$s (left); final result (fragment) of optimal approximation with optimal number of segments after 20 iterations; $E_{20} = 124093$, $T_{20} = 22$ s (right). The vector data reduction ratio is 5:1 (N=38924, M=7784). Processing time for full search algorithm is 157 s.

3.7.4 Summary

We have introduced two algorithms for solving the problem based on dynamic programming: full search and iterative reduced search. The algorithms optimize the number of segments and the approximation of the individual objects jointly. Experimental results indicate that the proposed algorithm reaches the optimal solution in all cases tested even though the optimality cannot be guaranteed in general. The iterative reduced search algorithm has time complexity of $O(N)$–$O(N^2)$ depending on the given the data reduction ratio.