PERGAMON

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Pattern Recognition 38 (2005) 381 – 394

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Data reduction of large vector graphics

Alexander Kolesnikov[a, b, *], Pasi Fränti[b]

[a] *Institute of Automation and Electrometry, SB RAS, Pr. Ak. Koptyuga, 1, 630090, Novosibirsk-90, Russia*
[b] *Department of Computer Science, University of Joensuu, P.O. Box 111, Joensuu, FI-80101, Finland*

Received 11 June 2003; received in revised form 24 February 2004; accepted 16 July 2004

Abstract

Fast algorithm for joint near-optimal approximation of multiple polygonal curves is proposed. It is based on iterative reduced search dynamic programming introduced earlier for the *min-ε problem* of a single polygonal curve. The proposed algorithm jointly optimizes the number of line segments allocated to the different individual curves, and the approximation of the curves by the given number of segments. Trade-off between time and optimality is controlled by the breadth of the search, and by the numbers of iterations applied.

Keywords: Polygonal approximation; Multiple-object; Dynamic programming; Min-ε problem; Data reduction; Vector data compression

1. Introduction

Approximation of *polygonal curves* is a classical problem in image processing, pattern recognition, computer graphics, digital cartography, and vector data processing. Optimal approximation of a single polygonal curve can be solved by methods from *graph theory* [1–5], *dynamic programming* [6–12], or *A*-search* [13,14] in $O(N^2)$–$O(N^3)$ time where N is the number of vertices in the input curve.

Faster but sub-optimal heuristics also exist with time complexities of $O(N)$–$O(N^2)$ [15,16]. Heuristic approaches for the approximation problem include *Split* [17–19,26], *Merge* [20–26], *Split-and-Merge* [27,28], *dominant points detection* [29–32], *sequential tracing* [33–35], *genetic algorithms* [36–39], *tabu search* [39,40], *ant colony methods* [41,42]. The case of closed contours includes also the optimal selection of the starting point. This can be solved by considering all input points and choosing the one with minimal error [8], by algorithm for all shortest paths in graph [3] or by heuristic approaches [2,9,43–45].

The polygonal approximation of a single curve can be extended to the case of multiple curves:

(a) *Multiple object min-# problem*: Given K polygonal curves $P_1, P_2, \ldots, P_K$, approximate it by K polygonal curves $Q_1, Q_2, \ldots, Q_K$ with the minimum total number of segments M so that the approximation error does not exceed a given maximum tolerance Δ.
(b) *Multiple object min-ε problem*: Given K polygonal curves $P_1, P_2, \ldots, P_K$, approximate it by K polygonal curves $Q_1, Q_2, \ldots, Q_K$ with a given total number of segments M so that the total approximation error is minimized. 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 [46]. In the case of additive error measures (L_1, L_2, etc.), on the other hand, the problem is not trivial [46]. Fortunately, in practical applications we mostly have to deal with error measure L_∞ in the case of min-# *problem*.

* Corresponding author.
E-mail address: koles@cs.joensuu.fi (A. Kolesnikov).

0031-3203/$30.00 © 2004 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2004.07.005

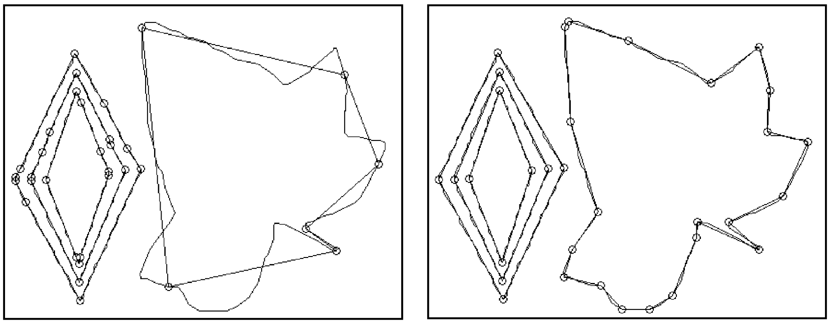

Fig. 1. Example of optimal approximation of multiple object with proportional distribution of the segments number ($M_k \approx N_k M/N$) (left), and with optimal distribution of the segments number (right). The number of points in the objects are $N_{\mathrm{D}} = 3 \times 121$ ("Diamond"), and $N_{\mathrm{L}} = 82$ ("Leaf"). The corresponding number of segments are $M_{\mathrm{D}} = 3 \times 9$ and $M_{\mathrm{L}} = 6$ with proportional distribution of the segments number, and $M_{\mathrm{D}} = 3 \times 4$ and $M_{\mathrm{L}} = 21$ with the optimal distribution of the segments number.

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. 1.

In literature, relatively little attention has been paid to the case of multi-object *min-*ε approximation even though it is far from trivial to solve it efficiently. The optimal solution have been introduced by Schuster and Katsaggelos [46] but the algorithm has time complexity of $O(N^2)$–$O(N^3)$ depending on the number of segments. This can be suitable for the encoding of object contours for MPEG-4 standard [47] but it is too slow in the case of large vector maps.

In this paper, we first generalize the dynamic programming approach of single object *min-*ε *problem* for the case of multiple objects. We then introduce a fast iterative reduced search algorithm based on the near-optimal approximation algorithm for the case of single object [48]. The proposed algorithm solves the approximation of the individual objects and the allocation of the segments jointly. 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. Moreover, the algorithm is significantly faster than the optimal counterpart; the time complexity is between $O(N)$–$O(N^2)$.

The rest of the paper is organized as follows. In Section 2, we recall the full search and the reduced search dynamic programming algorithms for the single-object problem. In Section 3, we generalize the dynamic programming approach for the case of multiple objects, and then introduce the iterative reduced search algorithm. Experiments and discussions are made in Section 4, and conclusions are drawn in Section 5.

2. Min-ε problem for single curve

Let us at first consider the optimal solution of the *min-*ε *problem* for single curve by dynamic programming algorithm proposed by Perez and Vidal [8]. We then recall the iterative reduced search approach introduced earlier in Ref. [48]. The proposed approach algorithm will then be generalized in the next sections for the approximation of multiple objects.

2.1. Problem formulation

An open *N-vertex polygonal curve P* in two-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 single object *min-*ε *problem* is stated as follows: approximate the polygonal curve P by another polygonal curve Q with a given number of linear segments M so that total approximation error $E(P, M)$ is minimized. The output 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. 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 $q_{m+1} = p_j$.

The error of approximation of curve segment $\{p_i, \ldots, p_j\}$ with the corresponding linear segment (q_m, q_{m+1}) is defined here as the sum of the squared Euclidean distances from each vertex of $\{p_i, \ldots, p_j\}$ to the correspondent line segment (q_m, q_{m+1}):

$$e^2(q_m, q_{m+1}) = \sum_{k=i+1}^{j-1} (y_k - a_{ij} x_k - b_{ij})^2 / (1 + a_{ij}^2), \quad (1)$$

where the coefficients a_{ij} and b_{ij} are defined from the linear equation $y = a_{ij} x + b_{ij}$ of the linear segment (p_i, p_j). The error $e^2(q_m, q_{m+1})$ with measure L_2 can be calculated in

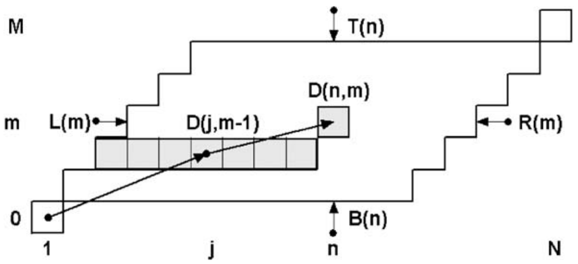

Fig. 2. Illustration of the single-goal state space Ω, and the dependencies of the calculation of the cost D for state (n, m) from the previous states $\{(j, m-1)\}$, where $j = L(m-1), \ldots, n-1$.

$O(1)$ time with five arrays of cumulatives of x, y, x^2, y^2, xy coordinates [8].

The total approximation error $E(P, M)$ of the input polygonal curve P by the output polygonal curve Q is the sum of the approximation errors of the curve segments $\{p_i, \ldots, p_j\}$ by the linear segments (q_m, q_{m+1}) for $m = 1, \ldots, M$:

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

To obtain optimal approximation we have to find the set of vertices $\{q_2, \ldots, q_M\}$ of Q that minimizes the cost function $E(P, M)$ for a given M:

$$E(P, M) = \min_{\{q_m\}} \sum_{m=1}^{M} e^2(q_m, q_{m+1}). \tag{3}$$

To solve the optimization task we first recall the dynamic programming algorithm [8].

2.2. Full search dynamic programming

Let us define two-dimensional discrete *state space* $\Omega = \{(n, m) : n = 1, \ldots, N; m = 0, \ldots, M\}$ as shown in Fig. 2. Every point (n, m) in the space Ω represents the sub-problem of the approximation of n-vertex polygonal curve $(p_1, \ldots, p_n)$ by m linear segments. The complete problem is represented by the goal state (N, M).

An approximation polygonal curve Q can be represented as a *path* $H(m)$ in the state space Ω from the start state $\Omega(1, 0)$ to the goal state (N, M). In the state space, we also define a function $D(n, m)$ of the state $\Omega(n, m)$ as the cost function value of the optimal approximation for the n-vertex polygonal curve $(p_1, \ldots, p_n)$ by m linear segments.

The state space Ω is bounded by left $L(m)$, right $R(m)$, bottom $B(n)$ and top $T(n)$ borders in the following way [48]:

$$L(m) = \begin{cases} m+1; & m = 0, 1, \ldots, M-1; \\ N; & m = M; \end{cases}$$

$$R(m) = \begin{cases} 1; & m = 0; \\ N - M + m; & m = 1, 2, \ldots, M. \end{cases}$$

$$B(n) = \begin{cases} 0; & n = 1; \\ 1; & n = 2, \ldots, N-M; \\ n - N + M; & n = N - M + 1, \ldots, N; \end{cases}$$

$$T(n) = \begin{cases} n-1 & n = 1, \ldots, M; \\ M-1; & n = M+1, \ldots, N-1; \\ M; & n = N. \end{cases} \tag{4}$$

The optimization problem can be solved by dynamic programming [8] in the bounded space (see Fig. 3) with the following recurrent equations:

$$D(n, m) = \min_{L(m-1) \leqslant j < n} \{D(j, m-1) + e^2(p_j, p_n)\},$$
$$A(n, m) = \arg\min_{L(m-1) \leqslant j < n} \{D(j, m-1) + e^2(p_j, p_n)\}, \tag{5}$$

where $n = 1, \ldots, N$ and $m = B(n), \ldots, T(n)$. 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). The time complexity of the algorithm is $O(MN^2)$, and the space complexity is $O(MN)$.

2.3. Iterative reduced search algorithm

Based on the dynamic programming we have introduced an iterative reduced search method [48]. This algorithm was intended to bridge the gap between slow but optimal, and fast but non-optimal heuristic algorithms. The algorithm includes the following three basic steps:

Step 1: Find *reference solution* with any fast heuristic algorithm. The obtained solution defines a reference path $H_0(m)$ in the state space Ω.

Step 2: Construct a single-goal bounding corridor of a fixed width W in the state space Ω along the reference path $H_0(m)$. The left $L(m)$, right $R(m)$, bottom $B(n)$, and top $T(n)$ bounds of the corridor (bounded state space) are defined in respect to the reference solution as follows:

$$L(m) = \begin{cases} m+1; & m = 0, \ldots, c_1, \\ \max\{m+1, H(m-c_1)\}; & m = c_1+1, \ldots, M, \end{cases}$$

$$R(m) = \begin{cases} \min\{N, H(m+c_2)-1\}, & m = 0, \ldots, M-c_2, \\ N; & m = M-c_2+1, \ldots, M, \end{cases} \tag{6}$$

```
IterativeReducedSearchDP(P, M);
REPEAT
   Q ← ReducedSearchDP(P, M);
UNTIL good enough

ReducedSearchDP(P, M);
D(1,0) ←0
FOR n = 2 TO N DO
   // a) Calculation of approximation errors
   FOR j = L(B(n) – 1) TO n–1 DO
      ν(n–j) ← e²(pj,pn)
   ENDFOR

   // b) Minimum search
   FOR m = B(n) TO T(n) DO
      dmin ← ∞
      FOR j= L(m–1) TO n–1 DO
         d ← D(j, m–1–B(j)) + ν(n–j)
         IF(d<dmin)
            dmin ← d;
            jmin ← j
         ENDIF
      ENDFOR
      D(n, m–B(n)) ← dmin
      A(n, m–B(n)) ← jmin
   ENDFOR
ENDFOR

// Restoration of the solution H1(m)
H1(M) = N
FOR M TO 1 DO
      H1(m–1) = A(H1(m), m – B(H1(m)))
ENDFOR
E ← D(N,M–B(N))
```

Fig. 3. General scheme of the iterative reduced search DP in the bounded state space.

$$B(n) = \begin{cases} 0; & n = 1, \\ m; & n = R(m-1)+1, \ldots, R(m); \end{cases}$$

$$T(n) = \begin{cases} \min\{M, m+W-1\}; & n = L(m), \ldots, L(m+1)-1; \\ M; & n = N. \end{cases}$$

where $c_1 = \lfloor W/2 \rfloor$, and $c_2 = W - c_1$ are the bounds of the corridor.

Step 3: Apply dynamic programming limited to the bounding corridor as shown in Fig. 3 with the recursive equations in Eq. (5).

These three steps are then iterated using the output solution $H_1(m)$ as a reference solution in the next iteration. Instead of the time consuming search in the full state space Ω the algorithm performs the search iteratively in the most relevant part of it. Trade-off between quality and time can be controlled by setting up the corridor width (W) appropriately, and by limiting the number of iterations (n_i). In Ref. [48], the optimal solutions were always found by setting up $W = 6$, and by iterating the algorithm until it converged. The pseudo code of the algorithm is given in Fig. 3.

The time complexity of the algorithm with n_i iterations is $O(n_i W^2 N^2/M)$, which varies between $O(N)$ and $O(N^2)$. The lower bound appears when M is large (proportional to N) and the upper bound when M is small (considered as constant). The speed-up in comparison to the full search is proportional to $(W/M)^2$. The space complexity of the algorithm is $O(WN)$.

3. Min-ϵ problem for multiple objects

We first formulate the multiple-objects *min-ε problem*, and then generalize the full search dynamic programming from the single object to the case of multiple objects. The iterative reduced search approach is then described.

3.1. Problem formulation

Consider the problem of joint approximation of *multiple* polygonal curves (objects), where we have K polygonal curves $P_1, \ldots, P_K$. The total number of vertices is $N = \Sigma N_k$, where N_k is the number of vertices in the object P_k. We have to approximate the set of polygonal curves by another set of polygonal curves $Q_1, \ldots, Q_K$. The total number of approximation line segments is ΣM_k, where M_k is the number of segments allocated to the approximation of a single polygonal curve Q_k.

The approximation *min-ε problem* for *multiple objects* can be formulated as follows: find the optimal approximation of the curves $P_1, \ldots, P_K$ by polygonal curves $Q_1, \ldots, Q_K$ with minimum error E under the given constraint on the total number of segments: $\Sigma M_k \leqslant M$.

The approximation error $E = E_k(P_k, M_k)$ of the input polygonal curve P_k by the output polygonal curve Q_k is the sum of the errors of the approximation of curve segments $\{p_{k,i}, \ldots, p_{k,j}\}$ of P_k by the line segments $(q_{k,m}, q_{k,m+1})$ of Q_k (see Eq. (2)):

$$E_k(P_k, M_k) = \sum_{m=1}^{M_k} e^2(q_{k,m}, q_{k,m+1}). \tag{7}$$

The total approximation error $E(P_1, \ldots, P_K, M)$ with measure L_2 is defined here as the sum of approximation errors for all objects P_k:

$$E(P_1, \ldots, P_K, M) = \sum_{k=1}^{K} E_k(P_k, M_k). \tag{8}$$

To obtain the optimal approximation of K objects we have to solve the following optimization task:

$$E(P_1, \ldots, P_K, M) = \min_{\{M_k\}} \min_{\{q_m\}} \sum_{k=1}^{K} \sum_{m=1}^{M_k-1} e^2(q_{k,m}, q_{k,m+1}), \quad \text{subject to } \sum_{k=1}^{K} M_k \leqslant M. \tag{9}$$

Two approaches have been proposed in Ref. [46] for the problem. The first approach is based on the Lagrangian multipliers method, which uses the DP algorithm for the shortest path in a directed acyclic graph. The second one is based on a tree-pruning algorithm. 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. The pruning-based approach 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 when N is relatively small. In the case of vector maps and digitized drawings, however, we have to process a large number of curves, and therefore, $O(N^2)$ can be too slow in practice.

3.2. Full search algorithm

Let us consider the cost (rate-distortion) function $g_k(M_k)$, which represents the approximation error for object P_k as a function of the number segments M_k:

$$g_k(M_k) = \min_{\{q_{k,m}\}} \sum_{m=1}^{M_k-1} e^2(q_{k,m}, q_{k,m+1}) \quad \text{where } M_k = 1, \ldots, \min\{M, N_k - 1\}. \tag{10}$$

The optimization task for the approximation error can be rewritten using the cost functions $g_k(M_k)$ as follows:

$$E(P_1, \ldots, P_K, M) = \min_{\{M_k\}} \sum_{k=1}^{K} g_k(M_k) \quad \text{subject to } \sum_{k=1}^{K} M_k \leqslant M. \tag{11}$$

The approximation problem for *multiple objects* differs from that of the *single object* problem in the following: in addition to the minimization of the individual objects we have to find the optimal numbers of segments M_k allocated to the objects $\{P_1, \ldots, P_K\}$.

The joint optimization problem can be solved by three step dynamic programming approach as follows:

Step 1: Solve the optimal approximation of every object by *multiple-goal* dynamic programming in order to obtain the cost functions $\{g_k(M_k)\}$;

Step 2: Solve the optimal allocation of the number of segments among the objects using the cost functions given by Step 1;

Step 3: Re-solve the optimal approximation of every object using the number of segments given by Step 2.

In step 1, we solve the optimal approximation of every object P_k using multiple-goal state space Ω_k as shown in Fig. 4 (left). In other words, we solve rate-distortion function $g_k(M_k)$ as the minimum approximation error of the object P_k with all possible number of segments M_k in the range $[1, \min\{M, N_k - 1\}]$. The bounds of the state space are defined as follows:

$$L_k(m) = \begin{cases} m+1; & m = 0, \ldots, M_k - 1; \\ N_k; & m = M_k; \end{cases}$$

$$R_k(m) = \begin{cases} 1; & m = 0; \\ N_k; & m = 1, \ldots, M_k. \end{cases}$$

$$T_k(m) = \begin{cases} m+1; & n = 0, \ldots, M_k - 2; \\ M_k - 1; & n = M_k - 1, \ldots, N_k - 2; \\ M_k; & n = N_k - 1; \end{cases}$$

$$B_k(m) = \begin{cases} 0; & n = 0; \\ 1; & n = 1, \ldots, N_k - 1. \end{cases} \tag{12}$$

In step 2, the optimal allocation of the segments $M_k^{(\mathrm{opt})}$ is found in order to minimize the total approximation error $E(P_1, \ldots, P_K, M)$. Let us consider the function $G_k(m)$ as the minimum approximation error of k objects with the total number of m segments:

$$G_k(m) = E(P_1, \ldots, P_k, m). \tag{13}$$

The problem of the optimal allocation of the constrained resource $\{M_k\}$ among the K objects can be solved by dynamic programming method with the following recursive equations [49] for the given functions $\{g_k(M_k)\}$:

$$G_k(m) = \min_{1 \leqslant x < N_k} \{g_k(x) + G_{k-1}(m-x)\}, \quad \text{where } m = 1, \ldots, \min\left\{M, \sum_{i=1}^{k-1} (N_i - 1)\right\}. \tag{14}$$

The function $G_1(m)$ for one object ($k = 1$) is given as follows: $G_1(m) = g_1(m)$, where $m = 1, \ldots, \min\{M, N_1 - 1\}$. The error of the optimal approximation of K objects with M segments is given as $E(P_1, \ldots, P_K, M) = G_K(M)$.

In step 3, we solve the optimal solution $H_k(m)$ for every object P_k with the found optimal number of segments $M_k^{(\mathrm{opt})}$. The optimal solutions are solved by the same DP

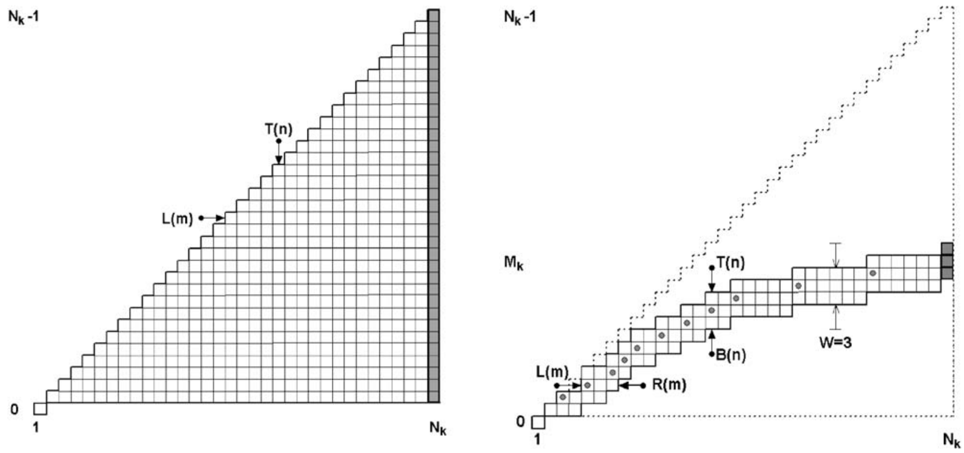

Fig. 4. Illustration of the *multiple-goal* state space Ω_k for sample problem of $N_k = 34$ (left), and the *multiple-goal* bounding corridor for sample problem of $N_k = 34$ and $M_k = 12$ using corridor width $W = 3$ (right). The reference path $H(m)$ is marked with dark gray circles, and the goal states with gray squares.

algorithm as applied in the first step but now with the fixed numbers of segments $M_k^{(\mathrm{opt})}$ given by the second step.

The time complexity of the first step is $O(N_k^3)$ for one object, and $O(\Sigma N_k^3)$ for all objects. This sums up to $O(N^3)$ in the worst case. The time complexity of the second step is $O(KM^2)$. The time complexity of the third step is $O(M_k^{(\mathrm{opt})} N_k^2)$ for one object, and $O(\Sigma\, M_k^{(\mathrm{opt})} N_k^2)$ for all objects. This sums up to $O(MN^2)$ in the worst case. The time complexity of the whole algorithm is dominated by the complexity of the first step, and is therefore $O(N^3)$.

The space complexity of the first step is determined by the memory requirement of the full search DP algorithm for the approximation of the biggest object: $\max\{N_k \times N_k\}$, which is $O(N^2)$ in the worst case. The space complexity of the second step with dynamic programming procedure is $O(K \times M)$. The memory requirement of the third step is defined by the memory needed for approximating the biggest object with the found optimal number of segments: $\max\{M_k^{(\mathrm{opt})} \times N_k\}$. The total space complexity of the algorithm is therefore determined by the complexity of the first step, which is $O(N^2)$.

3.3. Iterative reduced search algorithm

The full search DP algorithm introduced in Section 3.2 has the following drawbacks:

- The time complexity of the algorithm is $O(N^3)$, which can be too much for vector data with long curves of thousands of vertices.
- The memory requirements of the algorithm is $O(N^2)$. This can also be a limiting factor for processing of large vector maps with long curves.

We next generalize the iterative reduced search 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. The main difference to the full search is that a smaller search area is needed, which makes the algorithm faster. It also eliminates the need of the third step because of smaller memory requirements.

The algorithm for multiple-object *min-ε problem* with reduced search consists of the following steps (Fig. 5):

Step 1: Find preliminary approximation of every object for given initial number of segments;
Step 2: Iterate the following:

(a) Apply *multiple-goal reduced search* dynamic programming for the previous solution to define the cost functions $g_k(M_k)$;
(b) Solve the optimal allocation of the number of segments among the objects using the cost functions $g_k(M_k)$.

In step 1, we find a set of reference solutions $\{H_k(m)\}$ for every object P_k using some fast sub-optimal algorithm to distribute segments among the objects and perform polygonal approximation with the found number of segments. In this work, we use heuristic algorithms based on *Split* [17–19] and *Merge* [20–26] approaches, and random initialization. Any other fast heuristic algorithm can be used to obtain an initial solution.

In step 2a, multiple-goal state space Ω_k is constructed for each object with the following goal states: $M_k \in [a_k, b_k]$, where $a_k = \max\{1, M_k^{(0)} - c_1\}$, $b_k = \min\{M_k^{(0)} + c_2,\ M^{(0)}, N_k - 1\}$, and $c_1 = \lfloor W/2 \rfloor$, $c_2 = W - c_1$. Each state space Ω_k is then processed by the reduced

Algorithm for multiple-objects *min-ε problem*

// Step 1: Preliminary approximation
$\{Q_k\} \leftarrow$ FindPreliminaryApproximation($\{P_k\}$);
$\{M_k^{(0)}\} \leftarrow \{Q_k\}$

// Step 2: Iterative search
i ← 1;
REPEAT
  // Approximation of the objects
  FOR k = 1 TO K DO
    ReducedSearchDP(P_k, $M_k^{(i)}$);
    $g_k^{(i)} \leftarrow$ CostFunction(P_k, $M_k^{(i)}$);
  ENDFOR

  // Allocate resource
  $\{M_k^{(i+1)}\} \leftarrow$ ResourceAllocation($\{g_k^{(i)}(m)\}$, $\{M_k^{(i)}\}$);
  $Q_k \leftarrow H(M_k^{(j)})$
  i ← i+1;
UNTIL no changes

Fig. 5. Iterative reduced search algorithm for the *multiple object min-ε problem*.

search algorithm using revised bounding corridor of width $W_k = b_k - a_k + 1 \leqslant W$. The result of the search is W_k solutions $\{H_k(m)\}$ with the corresponding rate-distortion function $g_k(M_k)$ in the range $M_k \in [a_k, b_k]$. If the corridor width W_k is small ($W \leqslant 32$), the found paths $\{H_k^{(1)}(m)\}$ are stored in one-dimensional array of size N_k in order to avoid recalculation of the solutions later.

The left $L_k(m)$, right $R_k(m)$, bottom $B_k(n)$ and top $T_k(n)$ bounds of the multiple-goal bounding corridor are defined as follows:

$$L_k(m) = \begin{cases} m+1; & m = 0, \ldots, c_1, \\ \max\{m+1, H_k(m-c_1)\}; & m = c_1+1, \ldots, M_k, \end{cases}$$

$$R_k(m) = \begin{cases} \min\{N_k, H_k(m+c_2)-1\}; & m = 0, \ldots, M_k-c_2, \\ N_k; & m = M_k-c_2+1, \ldots, M_k, \end{cases} \tag{15}$$

$$B_k(n) = \begin{cases} 0; & n = 0, \\ m; & n = R_k(m-1)+1, \ldots, R_k(m), \end{cases}$$

$$B_k(n) = \begin{cases} m+W_k-1; & n = L_k(m), \ldots, L_k(m+1)-1; \\ M_k + W_k - 1; & n = N_k. \end{cases}$$

In step 2b, we find for every object P_k the optimal number of segments M_k in the range $[a_k, b_k]$. The optimal allocation of the constrained resource $\{M_k\}$ among the K objects $P_1, \ldots, P_K$ with the given cost functions $\{g_k(M_k)\}$ can be solved by dynamic programming with the following recursive expression ($k = 1, \ldots, K$):

$$G_k(m) = \min_{a_k \leqslant x \leqslant b_k} \{g_k(x) + G_{k-1}(m-x)\},$$
$$\text{where } m = \sum_{i=1}^{k-1} a_i, \ldots, \sum_{i=1}^{k-1} b_i. \tag{16}$$

The required value of the approximation error for K objects by M linear segments is defined from the cost function $G_k(m)$ as follows: $E(P_1, \ldots, P_K, M) = G_K(M)$. Finally, for every object P_k we restore the optimal solution $H_k(m)$ with the found number of segments $M_k^{(1)}$ from the stored paths $\{H_k(m)\}$.

The found numbers of segments $\{M_k^{(1)}\}$ are restricted to the range $[a_k, b_k]$, and they can provide only local minimum of the approximation error $E(P_1, \ldots, P_K, M)$. To find the global optimal allocation of the resource $\{M_k^{(\text{opt})}\}$ for the whole range of segments number, the iterations are necessary. The output solution of the previous iteration is used as the reference solution in the next iteration. Steps 2a and 2b are repeated until no changes appear in the approximation error values $G_K(M)$. The number of iterations depends on the bounding corridor width and how close the initial distribution of segments number $\{M_k^{(0)}\}$ is to the optimal distribution $\{M_k^{(\text{opt})}\}$.

While we iterate the algorithm to find the optimal distribution of the segments number M_k, we simultaneously optimize the location of the approximation vertices $\{q_{k,m}\}$ for the current number of segments M_k. Finally, the algorithm converges to approximation solution for all objects $\{P_1, \ldots, P_K\}$.

The time complexity of the algorithm is dominated by the first step. The processing time is $\Sigma(W_k^2 N_k^2 / M_k)$ in comparison to $\Sigma(N_k^3)$ of the full search. This can be roughly estimated as $O(W^2 N^2 / M)$, which varies from $O(N)$ to $O(N^2)$ depending on M. The processing time for the second step is reduced by a factor of $O(W/M)^2$ from the full search because the search range is reduced from M to W. The time complexity of the second step is $O(KW^2)$ in comparison to $O(KM^2)$ of the full search. At the third step, we restore the optimal solutions for the found number of segments from the stored paths. The time complexity of this simple procedure is $O(N)$.

To sum up, the time complexity of the reduced search algorithm for *multiple-object min-ε problem* is defined by the first step, and it is between $O(N)$ and $O(N^2)$. This is better than the $O(N^3)$ of the full search, and the $O(N^2 \log N)$ of the method proposed in Ref. [46].

The space complexity of the first step is reduced to $\max\{W \times N_k\}$ from $\max\{N_k^2\}$ of the full search as $W \ll N_k$. The memory requirement of the second step is also reduced from $K \times M$ to $K \times W$. In the third step, no additional memory is needed for restoring the optimal paths. The total space complexity of the proposed algorithm is defined by the complexity of the first step, which is $O(WN)$.

3.4. Heuristic algorithms for reference solution

To study sensitivity of the algorithm to the quality of reference solution we consider three strategies for creating the initial (reference) solution: (a) the *Merge-L_2* algorithm, (b) the *Split-based* algorithm with proportional distribution of segments, and (c) *Random* initialization with random distribution. These algorithms provide good, satisfactory, and poor initialization, respectively.

(a) *Merge-L_2* algorithm: The greedy algorithm in use is a generalization of Merge approach [20–26] from a single object to the case of multiple objects. First, all vertices of the shapes are considered as approximating points. The total number of approximating vertices is then iteratively reduced by elimination of the vertex q_m with the smallest cost function value $C(q_m)$. The process is halted when the desired total number of approximating segments M is reached. The cost function $C(q_m)$ of the vertex q_m with the two adjacent line segments (q_{m-1}, q_m) and (q_m, q_{m+1}) is defined as the change in total approximation error after replacing these segments with one segment (q_{m-1}, q_{m+1}):

$$C(q_m) = d(q_{m-1}, q_{m+1}) - d(q_{m-1}, q_m) - d(q_m, q_{m+1}),$$

where $d(q_i, q_j)$ is approximation error for line segment (q_i, q_j). We use integral square error as a cost function. The complexity of the Merge-based algorithm is $O(N \log N)$ [23,25] if we use heap structure to store the current values of the cost function of all vertices for K objects, and the complexity of algorithm for calculation of the cost function is $O(1)$. The *Merge-L_2* algorithm provides good distribution of segments among the objects.

(b) *Split-based* algorithm: the total number of segments M is distributed uniformly proportional to the number of vertices N_k in each object to give the initial values $M_k^{(0)}$ of the numbers of segments: $M_k^{(0)} \approx N_k M/N$. Then every object is approximated by the algorithm of Douglas–Peucker–Ramer individually for the calculated number of segments $M_k^{(0)}$ to obtain reference solution for the objects. The complexity of the algorithm is $O(M_k N_k)$ for an N_k-vertex object.

(c) *Random* algorithm: we simply take $(M-1)$ randomly chosen points for all the curves jointly, and create the initial approximation from them.

3.5. Approximation of closed contours

In the case of closed contours, we have to optimize the selection of the starting points as well. It can be done with the near-optimal algorithm we introduced recently in Ref. [45]. The proposed algorithm is based on reduced search dynamic programming algorithm for open curves [48]. It performs approximation of a cyclically extended input contour of double-size and then makes analysis of the state space to select the best starting point.

The processing time is double to that of the approximation of the corresponding open curve. The efficiency of the approach depends on the characteristics of the contours to be approximated, the number of segments, and the initial location of the starting points. For smooth curves with big number of approximation segments and a reasonably good initial selection for the starting points the improvement of the approximation can be negligible. In the case of contours with sharp corners and small number of segments, however, it can be worth to reduce the approximation error at the cost of double processing time. The selection of the relevant strategy depends on task in the question, the properties of the vector data, and the time resources.

4. Results and discussion

In order to evaluate the quality of sub-optimal algorithms, Rosin [15,16] introduced a measure known as *fidelity* (F). It measures how good a given sub-optimal solution is in respect to the optimal approximation in terms of the approximation error:

$$F = \frac{E_{\min}}{E} \times 100\%. \tag{17}$$

We test the proposed methods using the shapes shown in Fig. 6. The first and second shapes are didactic examples of the single and multiple-object cases. The third shape contains elevation lines from a sample map somewhere in Finland [50], and the fourth one is a large-scale vector map of Europe.

4.1. Iterative reduced search for single object

The iterative reduced search [48] is first illustrated for the test shape #1 in Fig. 7. The preliminary approximation with $M = 100$ is made by the Douglas–Peucker algorithm, which is then improved by iterative reduced search algorithm with corridor width $W = 10$ (see Fig. 7). The fidelity of the initial solution is $F_0 = 41.1\%(T_0 = 0.05\,\text{s})$; fidelity of the solutions after the 1st and 2nd iterations of the reduced search are $F_1 = 97.9\%(T_1 = 0.7\,\text{s})$ and $F_2 = 100\%(T_2 = 1.4\,\text{s})$, respectively.

For reference approximation obtained with *Merge-L_2* algorithm, the fidelity is $F_0 = 53.5\%(T_0 = 0.02\,\text{s})$; fidelity of the solutions after the 1st and 2nd iterations are $F_1 = 99.8\%(T_1 = 0.7\,\text{s})$ and $F_2 = 100\%(T_2 = 1.4\,\text{s})$, respectively.

With the full search dynamic programming algorithm of Perez and Vidal [8] the optimal result for the same test shape is achieved in $T = 166\,\text{s}$, and with fast A^*-search algorithm of Salotti [13] the optimal result is achieved in $T = 62\,\text{s}$. In this and the following tests, we use Pentium III, 2.0 GHz.

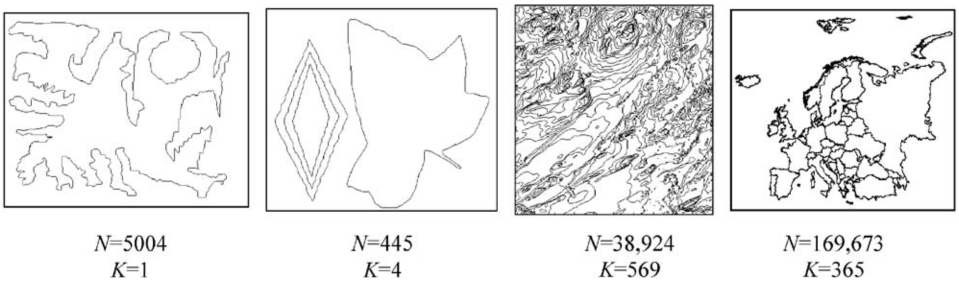

Fig. 6. Test data from left to right: Shape #1 is a digitized curve from [13]; #2: "diamond and leaf"; #3: elevation vector map; #4: vector map of Europe. Here N is the total number of points, and K is the number of objects. The images are available on web: `http://cs.joensuu.fi/pages/koles/images/`.

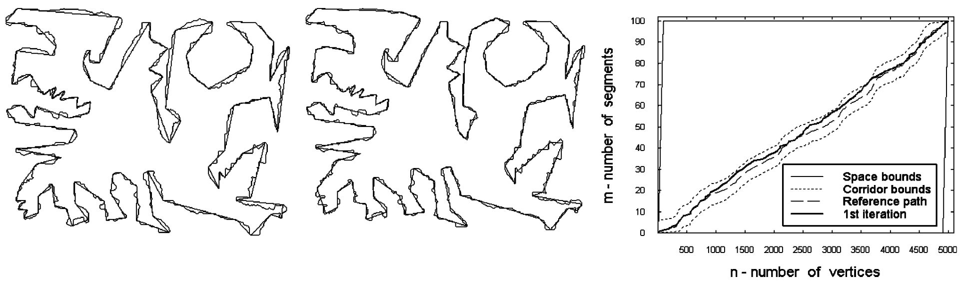

Fig. 7. Result of the approximation of test data #1 with $M = 100$ segments using Split-based algorithm (left), the iterative reduced search after the first iteration (middle), and the corresponding state space and the bounding corridor of width $W = 10$ (right).

4.2. Full search for multiple objects

The full search dynamic programming algorithm for the test data #2 is illustrated already in Fig. 1, which contain $N = 445$ vertices, and $M = 33$ linear approximation segments. With optimal allocation of the resources using the full search algorithm, the number of segments is $12(3 \times 4)$ in "Diamond", and 21 in "Leaf". The total approximation error is $E = 356$.

The test data #3 contain $N = 38,924$ vertices in $K = 569$ objects, and the approximation data contain $M = 7784$ linear segments (data reduction of 5:1). The processing time for the first step (calculation of the cost functions) is 69 s, the time for the resource allocation is 4 s, and the time for restoration of the optimal solutions is 14 s. In total, the processing time of the full search algorithm is 86 s.

The test data #4 consist of $K = 365$ shapes with $N = 169,673$ number of points. The data include several long curves up to 10,000 vertices. The approximation data contain $M = 8483$ linear segments corresponding to the reduction ratio of $N : M = 20 : 1$. Calculation of the result even for one 10,000-vertex object (finding 10,000 optimal solutions) with full search algorithm takes hours of computation. The memory requirements are also very high (about 600 Mbytes for the single 10,000-vertex curve). With the current hardware, we cannot perform the approximation of this data with the full search algorithm.

4.3. Iterative reduced search for multiple objects

At first we find approximation for the test data #3 (see Fig. 8). We consider three strategies for creating the initial distribution of the segments among the objects: (a) the *Merge-L_2* algorithm as described in Section 3.4, (b) the *Split-based* algorithm with proportional number of segments, and (c) *Random* initialization.

With the *Merge-L_2* algorithm, the fidelity of the initial solution is $F_0 = 71.6\%$ obtained in $T_0 = 0.3$ s. As the initial distribution of the segments is close to optimal, the reduced-search algorithm reaches very high fidelity of $F_1 = 99.95\%$ already after the first iteration in $T_1 = 1.3$ s. The final results ($F_3 = 100\%$) was achieved after three iterations in $T_3 = 2.8$ s instead of 86 s with the full search algorithm.

In the *Split-based* algorithm, the fidelity of the initial solution is $F_0 = 13.9\% (T_0 = 0.2\,\text{s})$. After one run of the optimization procedure with $W = 10$ the vector data is approximated with fidelity $F_1 = 50.3\%$ in $T_1 = 1.0$ s. The fidelity of $F_{15} = 99\%$ is reached after 15 iterations in $T_{15} = 12.0$ s, and the optimal result after 20 iterations in $T_{20} = 14.6$ s.

For obvious reasons, the fidelity of *Random* initial solution is very small ($F_0 = 0.3\%$). Nevertheless, optimal result ($F_{21} = 100\%$) was reached after 21 iterations in $T_{21} = 19.7$ s.

Next we find approximation of the test data #4 with iterative reduced search using corridor width $W = 10$ (see Fig. 9). As the algorithm converged to the same result with

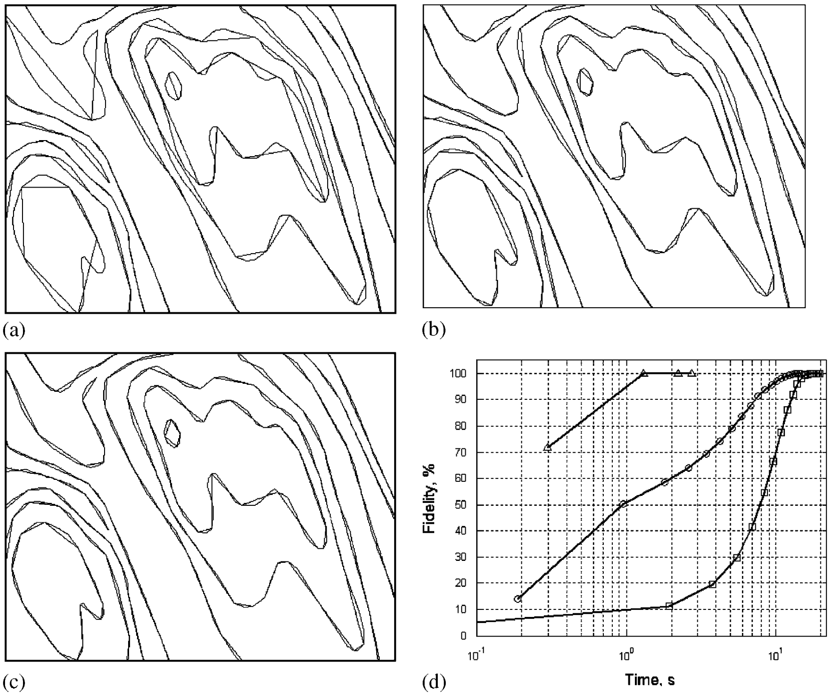

Fig. 8. Approximation results (fragment) for test data #3: (a) initial approximation with *Split-based* algorithm ($E = 892,158$); (b) initial approximation with *Merge-L_2* algorithm ($E = 173,362$); (c) final result ($E = 124,093$); (d) fidelity of the approximation as a function of time. The initial solution is obtained with *Merge-L_2* (triangles), *Split-based* (circles), and *Random* (squares) algorithm.

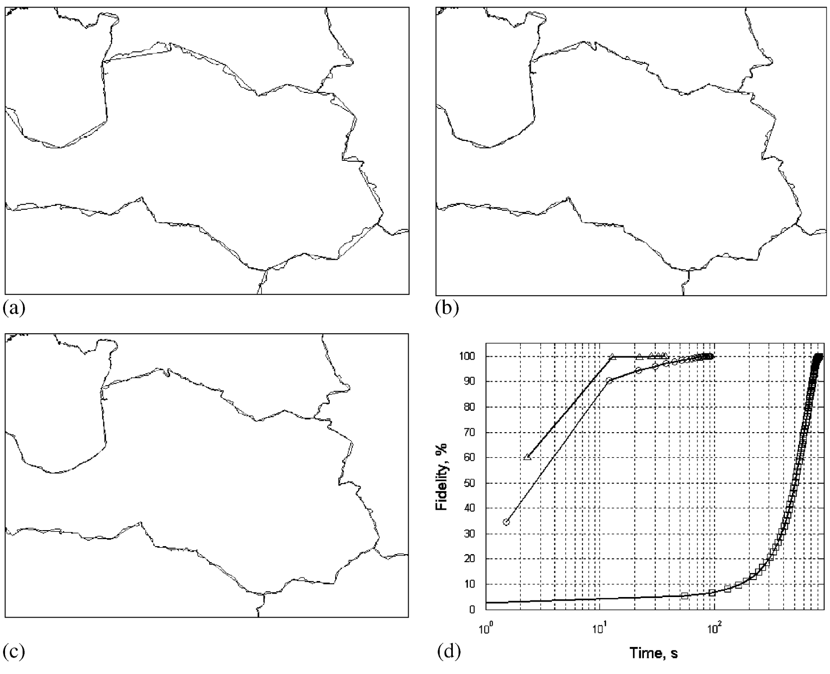

Fig. 9. Fragment of test data #4: (a) initial approximation with *Split-based* algorithm ($E = 57.12$); (b) initial approximation with *Merge-L_2* algorithm ($E = 32.89$); (c) the final result ($E = 19.76$); (d) fidelity of the approximation as a function of time. The initial solution is obtained with *Merge-L_2* (triangles), *Split-based* (circles), or *Random* (squares) algorithm.

Table 1
Fidelity and time (seconds) for the 1st iteration and final results for the test data #3 (a) and #4 (b). The initial approximation is obtained with *Merge-L_2* approach

	1st iteration		Final result		
	Fidelity (%)	Time (s)	Iterations	Fidelity (%)	Time (s)
(a) #3					
$W = 4$	98.4	0.6	8	99.7	2.3
$W = 6$	99.5	0.9	5	100	2.4
$W = 8$	99.8	1.1	4	100	2.7
$W = 10$	99.95	1.3	3	100	2.8
$W = 12$	99.99	1.6	3	100	3.3
$W = 14$	99.99	1.8	3	100	3.7
$W = 16$	100	2.1	2	100	3.8
$W = 18$	100	2.4	2	100	4.3
$W = 20$	100	2.7	2	100	4.8
(b) #4					
$W = 4$	97.5	5.6	24	99.5	19.9
$W = 6$	98.9	7.5	10	99.9	26.0
$W = 8$	99.4	10.0	8	100	29.9
$W = 10$	99.6	12.8	6	100	36.7
$W = 12$	99.7	16.2	5	100	43.8
$W = 14$	99.8	19.6	5	100	49.0
$W = 16$	99.9	23.3	5	100	57.3
$W = 18$	99.9	27.7	4	100	62.3
$W = 20$	99.9	32.1	3	100	72.1

Table 2
The minimum number of iterations and the corresponding run times in which the algorithm reaches certain fidelity level with the test data #3 (a) and #4 (b). The initial approximation is obtained with the *Split-based* algorithm

	90% fidelity		99% fidelity		Final result		
	Iterations	Time (s)	Iterations	Time (s)	Iterations	Time (s)	Fidelity (%)
(a) #3							
$W = 4$	36	10.3	58	16.0	76	19.5	99.5
$W = 6$	18	8.0	29	12.2	29	15.0	100
$W = 8$	12	7.5	19	11.6	26	14.4	100
$W = 10$	9	7.5	15	12.0	20	14.6	100
$W = 12$	7	7.2	12	12.1	20	16.5	100
$W = 14$	6	7.6	10	12.5	16	16.6	100
$W = 16$	5	7.4	9	13.2	16	18.0	100
$W = 20$	4	7.7	7	13.7	16	21.0	100
(b) #4							
$W = 4$	4	12	43	61	67	72	99.2
$W = 6$	2	11	18	49	35	62	99.4
$W = 8$	2	16	10	54	28	81	100
$W = 10$	1	11	8	62	22	92	100
$W = 12$	1	14	7	72	19	104	100
$W = 14$	1	19	6	99	16	116	100
$W = 16$	1	22	5	86	14	127	100
$W = 20$	1	31	4	103	12	156	100

Table 3
Summary of the fidelity and the processing times (seconds). The initial approximation is obtained with *Merge-L_2* algorithm; corridor width $W = 10$

Set	N	K	M	Merge-L$_2$		Reduced search		Full search	
				Fidelity (%)	Time (s)	Fidelity	Time (s)	Fidelity (%)	Time (s)
#1	5004	1	100	53.5	0.02	100	2.1	100	169.2
#2	445	4	33	82.5	< 0.01	100	0.02	100	0.03
#3	38 924	569	7784	71.6	0.30	100	2.8	100	86.0
#4	169 673	365	8483	59.9	2.30	≈ 100	36.9	N/A	N/A

Table 4
Summary of the fidelity and the processing times (seconds). The initial approximation is obtained with *Split-based* algorithm; corridor width $W = 10$

Set	N	K	M	Split-based		Reduced search		Full search	
				Fidelity (%)	Time (s)	Fidelity (%)	Time (s)	Fidelity	Time (s)
#1	5004	1	100	41.1	0.03	100	2.1	100	169.2
#2	445	4	33	5.0	< 0.01	100	0.06	100	0.03
#3	38 924	569	7784	13.9	0.19	100	15.0	100	86.0
#4	169 673	365	8483	34.6	1.50	≈ 100	92.7	N/A	N/A

Table 5
Summary of the fidelity and the processing times (seconds). The initial approximation is obtained with *Random* algorithm; corridor width $W = 10$

Set	N	K	M	Random		Reduced search		Full search	
				Fidelity (%)	Time (s)	Fidelity (%)	Time (s)	Fidelity	Time (s)
#1	5004	1	100	2.4	< 0.01	100	2.1	100	169.2
#2	445	4	33	0.2	< 0.01	100	0.02	100	0.03
#3	38 924	569	7784	0.3	< 0.01	100	19.7	100	86.0
#4	169 673	365	8483	0.1	< 0.01	≈ 100	829.0	N/A	N/A

all parameter values $W = 8$–32, we expect that it is also the optimal solution.

With the *Merge-L_2* approach, the fidelity of the initial solution is $F_0 = 59.9\%(T_0 = 2.3\,\mathrm{s})$. The fidelity after the 1st iteration is $F_1 = 99.6\%$ in $T_1 = 12.8\,\mathrm{s}$, and the final result ($F_6 = 100\%$) was achieved after six iterations in $T_6 = 36.9\,\mathrm{s}$. Since the solution of the full search algorithm is not available, the fidelity is calculated in these cases relative to the best solution found.

For the *Split-based* algorithm, the fidelity of the initial solution is $F_0 = 34.6\%(T_0 = 1.5\,\mathrm{s})$. The fidelity $F_1 = 89.5\%$ was reached ($T_1 = 12\,\mathrm{s}$) after the first iteration, and near-optimal result with the fidelity of $F_8 = 99\%$ was achieved after 8 iterations ($T_8 = 68\,\mathrm{s}$). The final solution was obtained after 22 iterations ($T_{22} = 93\,\mathrm{s}$).

The fidelity of *Random* initial solution is $F_0 = 0.1\%$. It takes 110 iterations and $T_{110} = 829\,\mathrm{s}$ to converge to the same approximation as with the *Split-based* and *Merge-L_2* methods.

From comparisons of the three different initialization strategies we can see that the number of iterations and processing time mostly depends on how close the initial distribution is to the optimal one. The quality of the approximation of a single object is not so critical because it can be greatly improved by 2-4 runs of iterative reduced search algorithm [48] applied to the objects individually for the current distribution.

The effect of the corridor width W is reported in Tables 1 and 2 as the number of iterations (and running time, respectively) needed to obtain approximation with fidelity of 90, 99%, and the final result, respectively. The use of a wider corridor increases the processing time of a single iteration but, at the same time, decreases the total number of iterations needed. The overall results are roughly equal for most of the parameter values tested in respect to the time-distortion performance. The exceptions are the smallest parameter values ($W = 4$–6), which do not always result in the optimal solution although quite close anyway ($F > 99\%$).

On the basis of the results, we recommend parameter value $W=10$ and conclude, that the exact choice of the parameter is not crucial for the performance of the algorithm.

The main results of the reduced search are summarized in Tables 3–5, and compared to that of the full search. Vector data with a moderate number of objects and vertices (Sets #1, #2 and #3) can also be processed with the full search but the reduced search is significantly faster. In the case of a very large data set, however, the memory requirements were too large and the approximation would have taken hours. In such case, the reduced search should be used.

From the results we can see that the proposed algorithm converges for $W>8$ to the optimal approximation (fidelity of 100%) no matter which initialization is used. The *Merge-*L_2 algorithm, however, provides better initial solution than the *Split*-based algorithm with comparable times because of better segments distribution. The consequence is that a smaller number of iterations is sufficient to reach the optimal solution, and thus, shorter overall running time is needed. We therefore conclude that the use of a better initialization can provide further speed-up in the case of large data sets. Nevertheless, any initialization is enough in order to reach the optimal approximation with the tested data sets.

5. Conclusions

In the paper, the *min-*ε *problem* of optimal approximation of multiple-object vector data was considered. The problem is treated as optimization task with approximation error as cost function. We 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 number of segments. This is significantly smaller than the $O(N^3)$ of the full search, or the $O(N^2 \log(N))$ of Ref. [42]. The reduced search approach is also applicable for very large data sets with reasonable memory requirements. The algorithm can also be tuned for obtaining fast sub-optimal solutions by reducing the number of iterations and corridor width.

With the tested data sets, the algorithm converged to the optimal solution with all three tested initializations, including random reference solution. The greedy Merge-based algorithm provided best initial solution, which was benefited as faster convergence than that of the other initializations. It is expected that any initial solution can be used with the proposed method, even though a higher quality initialization can save the number of iterations and processing time.

References

[1] H. Imai, M. Iri, Computational-geometric methods for polygonal approximations of a curve, Comput. Vision Image Process. 36 (1986) 31–41.

[2] A. Pikaz, I. Dinstein, Optimal polygonal approximation of digital curves, Pattern Recogn. 28 (3) (1995) 371–379.

[3] W.S. Chan, F. Chin, On approximation of polygonal curves with minimum number of line segments or minimum error, Int. J. Comput. Geom. Appl. 6 (1996) 59–77.

[4] D.Z. Chen, O. Daescu, Space-efficient algorithms for approximating polygonal curves in two-dimensional space, Int. J. Comput. Geom. Appl. 13 (2) (2003) 95–111.

[5] G. Barequet, D.Z. Chen, O. Daescu, M.T. Goodrich, J. Snoeyink, Efficiently approximating polygonal paths in three and higher dimensions, Algorithmica 33 (2) (2002) 150–167.

[6] G. Papakonstantinou, Optimal polygonal approximation of digital curves, Signal Process. 8 (1985) 131–135.

[7] J.G. Dunham, Optimum uniform piecewise linear approximation of planar curves, IEEE Trans. Pattern Anal. Mach. Intell. 8 (1) (1986) 67–75.

[8] J.C. Perez, E. Vidal, Optimum polygonal approximation of digitized curves, Pattern Recogn. Lett. 15 (1994) 743–750.

[9] Y. Zhu, L.D. Seneviratne, Optimal polygonal approximation of digitized curves, IEE Proc.-Vis. Image Signal Process. 144 (1) (1997) 8–14.

[10] C.-C. Tseng, C.-J. Juan, H.-C. Chang, J.-F. Lin, An optimal line segment extraction algorithm for online Chinese character recognition using dynamic programming, Pattern Recogn. Lett. 19 (1998) 953–961.

[11] R. Nygaard, J. Husøy, D. Haugland, Compression of image contours using combinatorial optimization, Proceedings of the International Conference on Image Processing-ICIP'98, vol. 1, 1998, pp. 266–270.

[12] M. Salotti, Un algorithme efficiace pour l'approximation polygonale optimale, 13$^{\text{ème}}$ Congrès Francophone AFRIF-AFIA de Reconnaissance des Formes et Intelligence Artificielle-RFIA'2002, Angers, France, vol. 1, 2002, pp. 11–18.

[13] M. Salotti, An efficient algorithm for the optimal polygonal approximation of digitized curves, Pattern Recogn. Lett. 22 (2001) 215–221.

[14] M. Salotti, Optimal polygonal approximation of digitized curves using the sum of square deviations criterion, Pattern Recogn. 35 (2) (2002) 435–443.

[15] P.L. Rosin, Techniques for assessing polygonal approximations of curves, IEEE Trans. Pattern Anal. Mach. Intell. 14 (6) (1997) 659–666.

[16] P.L. Rosin, Techniques for assessing of behaviour of polygonal approximations of curves, Pattern Recogn. 36 (6) (2003) 505–518.

[17] U. Ramer, An iterative procedure for polygonal approximation of plane curves, Comput. Graph. Image Process. 1 (1972) 244–256.

[18] D.H. Douglas, T.K. Peucker, Algorithm for the reduction of the number of points required to represent a line or its caricature, The Can. Cartograph. 10 (2) (1973) 112–122.

[19] J. Hershberger, J. Snoeyink, Cartografic line simplification and polygon CSG formulæin $O(n \log^* n)$ time, Comput. Geom. Theory Appl. 11 (3–4) (1998) 175–185.

[20] J.-S. Wu, J.-J. Leou, New polygonal approximation schemes for object shape representation, Pattern Recogn. 26 (1993) 471–484.
[21] M. Visvalingam, J. Whyatt, Line generalization by repeated elimination of points, Cartograph. J. 30 (1) (1993) 46–51.
[22] L. Boxer, C.-S. Chang, R. Miller, A. Rau-Chaplin, Polygonal approximation by boundary reduction, Pattern Recogn. Lett. 14 (1993) 111–119.
[23] A. Pikaz, I. Dinstein, An algorithm for polygonal approximation based on iterative point elimination, Pattern Recogn. Lett. 16 (6) (1995) 557–563.
[24] K.M. Ku, P.K. Chiu, Polygonal approximation of digital curve by graduate iterative merging, Electron. Lett. 31 (1995) 444–446.
[25] L.J. Latecki, R. Lakämper, Convexity rule for shape decomposition based on discrete contour evolution, Comput. Vision Image Understand. 73 (1999) 441–454.
[26] H.Z. Shu, L.M. Luo, J.D. Zhou, X.D. Bao, Moment-based methods for polygonal approximation of digitized curves, Pattern Recogn. 32 (2002) 421–434.
[27] T. Pavlidis, S.L. Horovitz, Segmentation of plane curves, IEEE Trans. Comput. 23 (8) (1974) 860–870.
[28] Y. Xiao, J.J. Zou, H. Yan, An adaptive split-and-merge method for binary image contour data compression, Pattern Recogn. Lett. 22 (2001) 299–307.
[29] C.-H. Teh, R.T. Chin, On the detection of dominant points on digital curves, IEEE Trans. Pattern Anal. Mach. Intell. 11 (8) (1989) 859–872.
[30] P. Zhu, P.M. Chirlian, On critical point detection of digital shapes, IEEE Trans. Pattern Anal. Mach. Intell. 17 (1995) 737–748.
[31] J.M. Iñesta, M. Buendia, M.A. Sarti, Reliable polygonal approximations of imaged real objects through dominant point detection, Pattern Recogn. 31 (1998) 685–697.
[32] A. Garrido, N.P. de la Blanca, M. Garcia-Silvente, Boundary simplification using a multiscale dominant-point detection algorithm, Pattern Recogn. 31 (1998) 791–804.
[33] J. Sklansky, V. Gonzalez, Fast polygonal approximation of digitized curves, Pattern Recogn. 12 (1980) 327–331.
[34] Y. Kurozumi, W.A. Davis, Polygonal approximation by the minimax method, Computer Vision, Graph. Image Process. 19 (1982) 248–264.
[35] B.K. Ray, K.S. Ray, A non-parametric sequential method for polygonal approximation of digital curves, Pattern Recogn. Lett. 15 (1994) 161–167.
[36] P.-Y. Yin, A new method for polygonal approximation using genetic algorithms, Pattern Recogn. Lett. 19 (1998) 1017–1026.
[37] S.-C. Huang, Y.-N. Sun, Polygonal approximation using genetic algorithms, Pattern Recogn. 32 (1999) 1017–1026.
[38] S.-Y. Ho, Y.-C. Chen, An efficient evolutionary algorithm for accurate polygonal approximation, Pattern Recogn. 34 (2001) 2305–2317.
[39] H. Zhang, J. Guo, Optimal polygonal approximation of digital planar curves using meta-heuristics, Pattern Recogn. 34 (2001) 1429–1436.
[40] P.-Y. Yin, A tabu search approach to the approximation of digital curves, Int. J. Pattern Recogn. Artif. Intell. 14 (2000) 243–255.
[41] U. Vallone, Bidimensional shapes polygonalization by ACO, in: Proceedings of the 3rd International Workshop on Ants Algorithms, Brussels, Belgium, 2002, pp. 296–297.
[42] P.-Y. Yin, Ant colony search algorithms for optimal approximation of plane curve, Pattern Recogn. 36 (2003) 1783–1797.
[43] K. Schroeder, P. Laurent, Efficient polygon approximations for shape signatures, Proceedings of the International Conference on Image Processing-ICIP'99, vol. 2, 1999, pp. 811–814.
[44] J.-H. Horng, J.T. Li, An automatic and efficient dynamic programming algorithm for polygonal approximation of digital curves, Pattern Recogn. Lett. 23 (2002) 171–182.
[45] A. Kolesnikov, P. Fränti, Polygonal approximation of closed curves, Lecture Notes in Computer Science, vol. 2749, pp. 778–785, Proceedings of the 12th Scandinavian Conference on Image Analysis-SCIA'2003, 2003, Göteborg, Sweden.
[46] G.M. Schuster, A.K. Katsaggelos, An optimal polygonal boundary encoding scheme in the rate-distortion sense, IEEE Trans. Image Process. 7 (1) (1998) 13–26.
[47] A.K. Katsaggelos, L.P. Kondi, F.W. Meier, J. Osterman, G.M. Schuster, MPEG-4 and rate-distortion-based shape-coding techniques, Proc. IEEE 86 (6) (1998) 1126–1154.
[48] A. Kolesnikov, P. Fränti, Reduced search dynamic programming for approximation of polygonal curves, Pattern Recogn. Lett. 24 (2003) 2243–2254.
[49] R. Bellman, Dynamic Programming, Princeton University Press, NJ, 1957.
[50] National Land Survey of Finland, Opastinsilta 12 C, P.O. Box 84, 00521 Helsinki, Finland, http://www.maanmittauslaitos.fi/.

About the Author—ALEXANDER KOLESNIKOV received the MSc degree in physics in 1976 from Novosibirsk State University, USSR, and Ph.D degree in computer science in 2003 from University of Joensuu, Finland. From 1976 to 2003 he was research fellow and senior research fellow with the Institute of Automation and Electrometry, Russian Academy of Sciences, Novosibirsk, Russia. Currently he is researcher in the Department of Computer Science with the University of Joensuu, Finland. His main research areas are in signal and image processing.

About the Author—PASI FRÄNTI received the MSc and Ph.D degrees in computer science in 1991 and 1994, respectively, from the University of Turku, Finland. From 1996 to 1999 he was a postdoctoral researcher with the University of Joensuu (funded by the Academy of Finland), where he has been a professor since 2000. His primary research interests are in image compression, vector quantization, and clustering algorithms.