

Rubberband algorithms for the efficient solution of various Euclidean shortest path problems

Fajie Li and Reinhard Klette

*The University of Auckland
New Zealand*

General Euclidean shortest path (ESP) problem

Given a Euclidean space which contains (closed) polyhedral obstacles; compute a path which

- (i) connects two given points in the space,
- (ii) does not intersect the interior of any obstacle,
- (iii) is of minimum Euclidean length.

Examples: ESP inside of a simple cube-arc or inside of a simple polygon, on the surface of a convex polytope, or inside of a simply-connected polyhedron, or problems such as touring polygons, parts cutting, safari or zookeeper, or the watchman route. All-together, a class of immensely important computational problems of huge impact in economy, science or technology.

Time complexities

For 3D (e.g., path-planning in robotics), general ESP algorithms are NP-hard (J. Canny and J. H. Reif 1987).

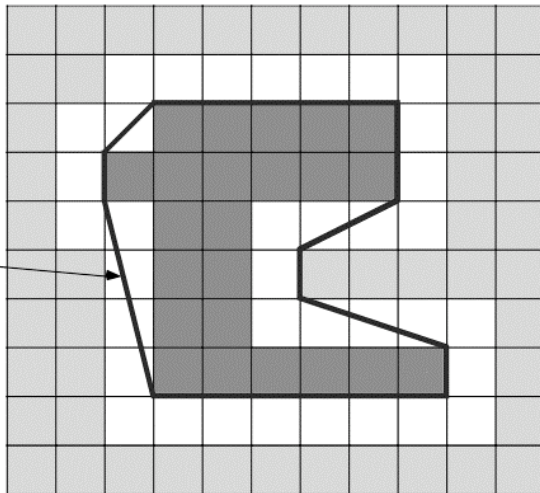
For 2D, there are linear-time, but very complicated algorithms (e.g., for ESP in simple polygon, based on a triangulation of the whole polygon, see
B. Chazelle, 1991)

or linear-time and easy-to-implement algorithms

(e.g., for the relative convex hull in the 2D grid, see

R. Klette, V. Kovalevsky, and B. Yip, 1999).

Relative
convex
hull



This talk is about RBAs, namely

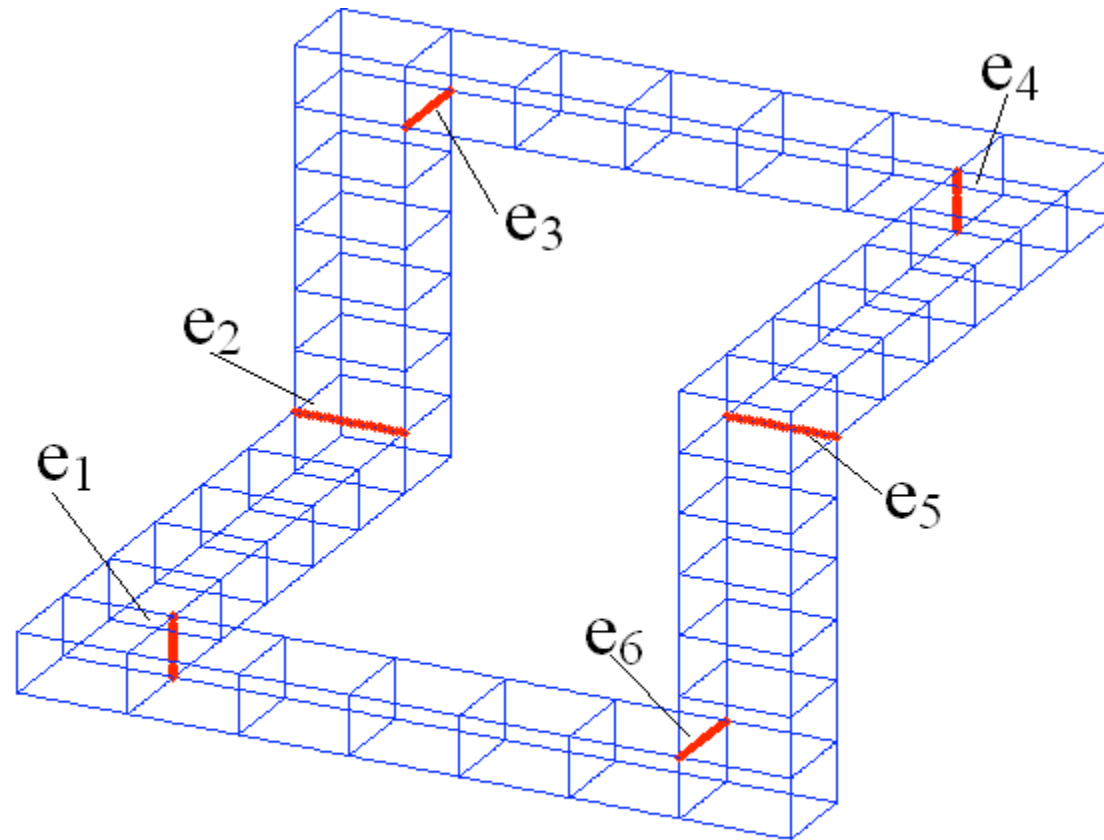
the original rubberband algorithm (RBA) for the calculation of Euclidean shortest paths (ESPs) in cube-curves in 3D space

(T. Buelow and R. Klette, 2000 - 2002),

the analysis of this algorithm, leading to two approximate algorithms, (an edge-based or face-based) corrected rubberband algorithms, and (MOST IMPORTANTLY) to generalizations of basic ideas for efficient solutions for 2D or 3D ESP problems by rubberband algorithms.

(F. Li and R. Klette, 2003 - 2006).

Critical edges in simple cube curves



critical edges: $e_1, e_2, e_3, e_4, e_5, e_6$

Critical edges are the only possible locations for vertices of an ESP (R. Klette and T. Buelow, DGCI 2000); a subset of those will define the *step set*.

Original RBA (ICPR 2000, IEEE PAMI 2002):

Check for Options 1, 2 or 3 in successive iterations n , until *change in length* L_n of curves is below ε

OP₁+ OP₂: find *step set* of critical edges $\{e_0, e_1, \dots, e_m\}$ such that e_i contains exactly one vertex of the path, for $i = 0, 1, \dots, m$.

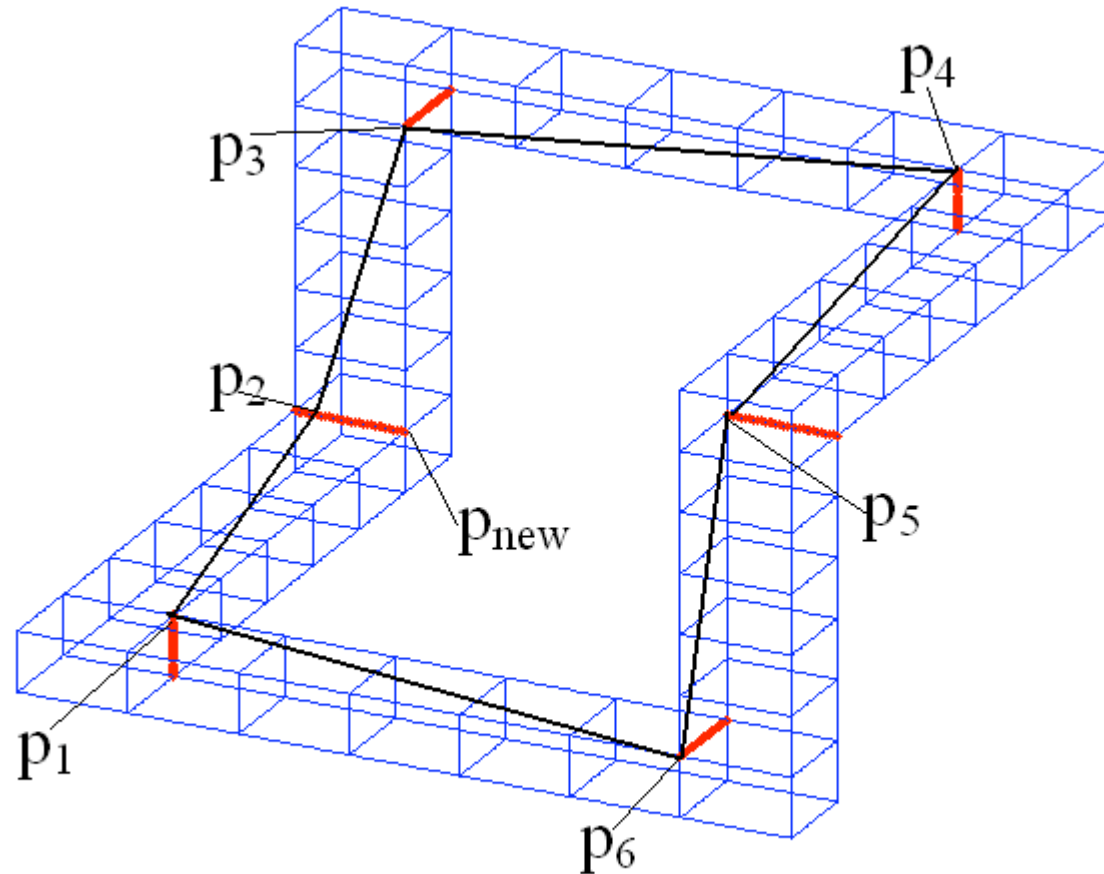
OP₃:

- move p_i on its critical edge e into optimum position

$$p_{\text{new}} \text{ with } |p_{\text{new}} - p_{i-1}| + |p_{i+1} - p_{\text{new}}| \\ = \min\{|p - p_{i-1}| + |p_{i+1} - p| : p \text{ in } e\}$$

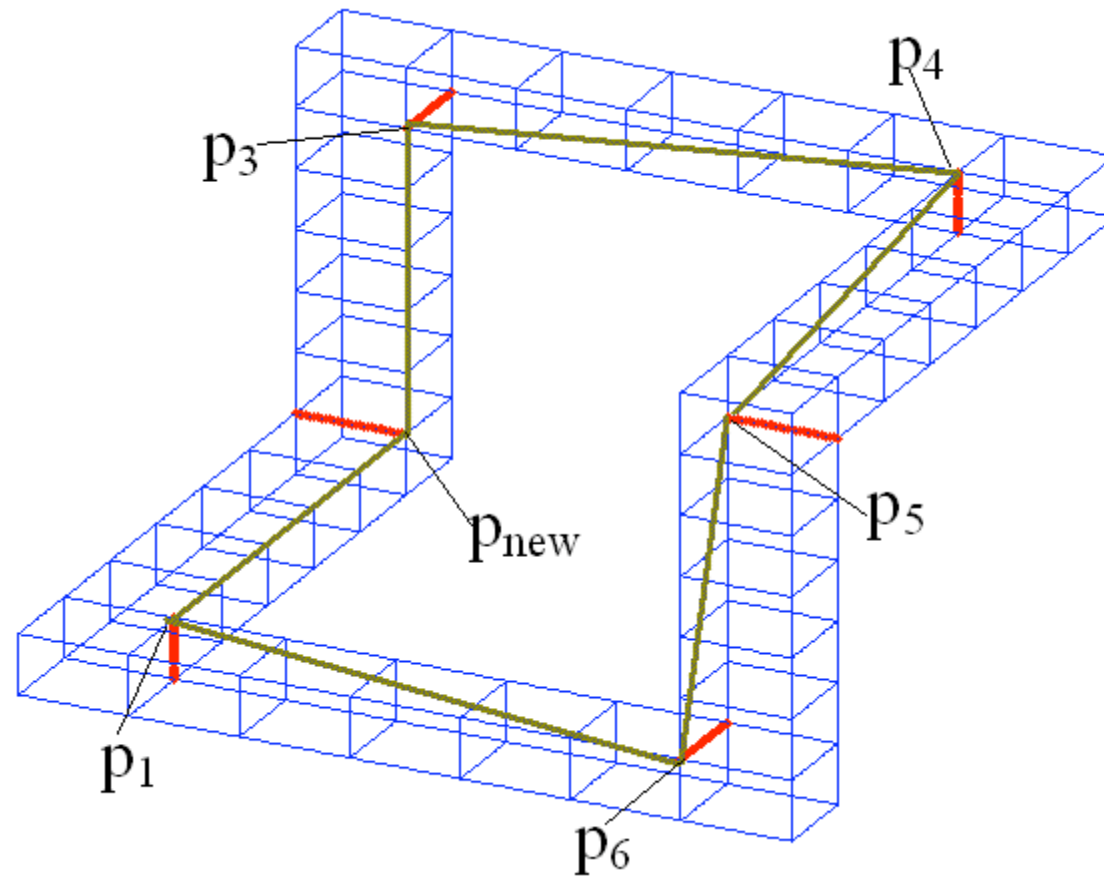
- replace (p_{i-1}, p_i, p_{i+1}) by $(p_{i-1}, p_{\text{new}}, p_{i+1})$ in path
- continue with vertices $p_{\text{new}}, p_{i+1}, p_{i+2}$.

Example of local optimization in OP_3



$$|p_{\text{new}} - p_1| + |p_3 - p_{\text{new}}| = \min \{|p - p_1| + |p_3 - p| : p \text{ in } e_2\}$$

An application of OP_3 : calculation of p_{new} .



Updated path.

Original RBA more in detail:

Algorithm which consists of two subprocesses:

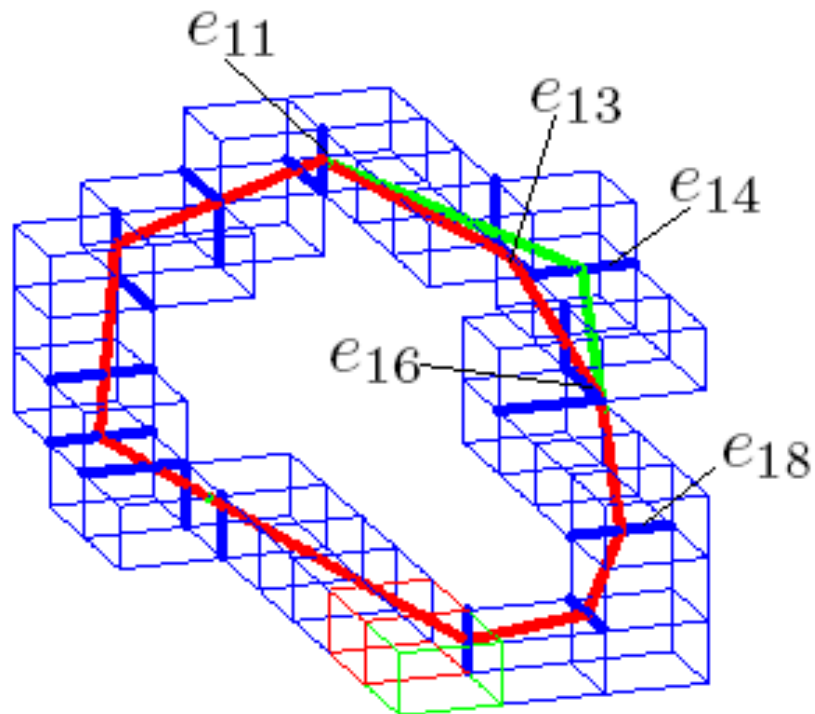
(i) an initialization process (e.g., from endpoint of critical edge to closest endpoint of next critical edge), and

(ii) an iterative process ($\mathbf{OP}_1 + \mathbf{OP}_2 + \mathbf{OP}_3$) which contracts the path in each step (using a *break-off criterion*: $L_n - L_{n+1} < \varepsilon$), with

\mathbf{OP}_1 : delete p_i if $p_{i-1}p_{i+1}$ is in *tube* (= union of all cubes), or

\mathbf{OP}_2 : calculate intersection points of triangle $p_{i-1} p_i p_{i+1}$ with critical edges and take resulting convex arc (in tube of curve) of intersection points

Illustration for (original) Option 2 :



ceIDs_1 =

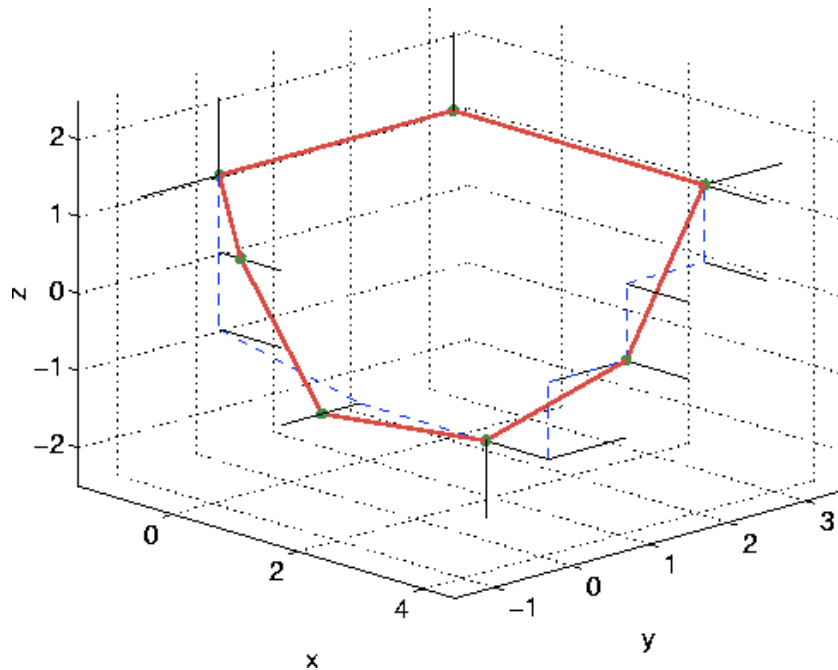
4 7 11 14 18 19 20

ceIDs_2 =

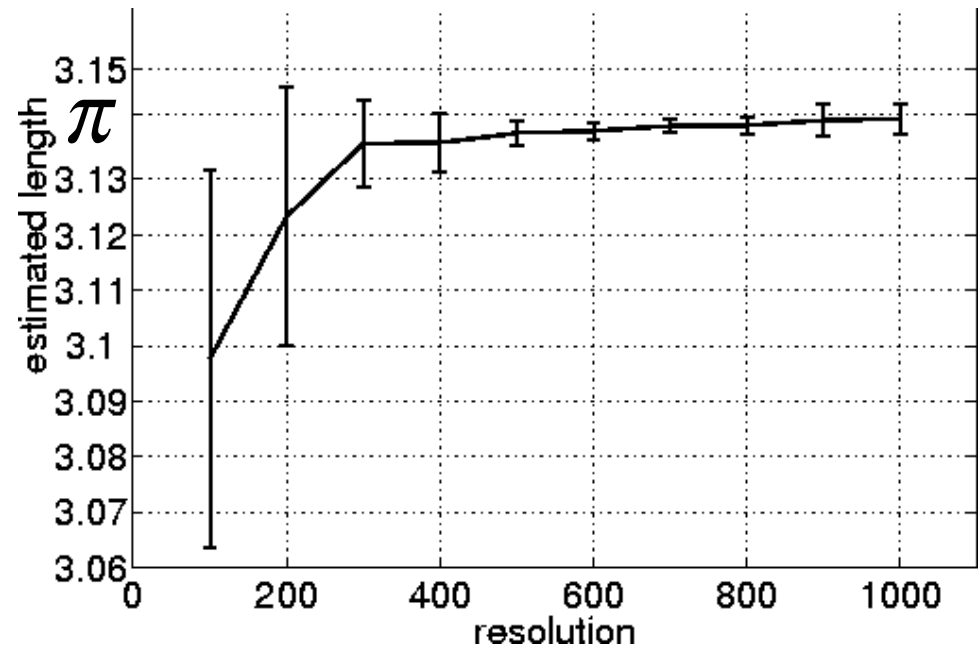
4 7 11 13 16 18 19 20

In general it may be 11,14,18 again in next iteration - of course, for decreasing length of calculated curve.

Experimental evaluation (2000-2002)



Critical edges (short black lines), initialization (dashed blue line) and final result (red line).



Results on digitized 3D curves with known length π , for different grid resolutions.

Situation with original RBA in 2002

Even for very small values of ε , the measured time complexity was $O(n)$, where n is the number of cubes on the given cube-curve. However, *no proof for asymptotic time complexity of original rubberband algorithm by 2002.*

For a small number of test examples, calculated paths seemed (!) to converge against ESP. However, no implemented algorithm for calculating the ESP, and (more general) *no proof whether path provided by original rubberband algorithm actually converges against ESP.*

(Algorithm in use since 2002, e.g. in DNA research).

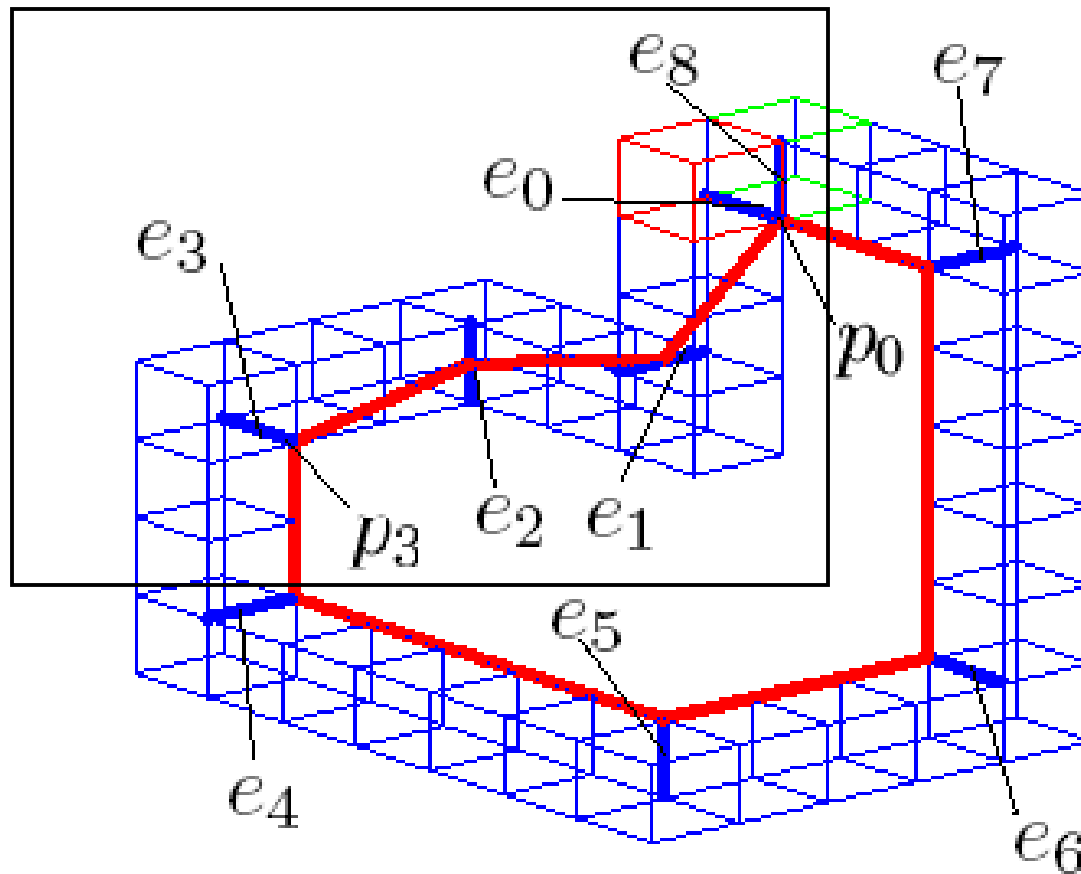
Arithmetic algorithms

An *arithmetic algorithm* consists of a finite number of steps of arithmetic operations, possibly also using input parameters from the field of rational numbers, using only the following basic operators: $+$, $-$, \cdot , $/$ or the k th root, for $k \geq 2$.

Example: The problem of finding the roots of $p(x) = 84x^6 - 228x^5 + 361x^4 + 20x^3 + 210x^2 - 200x + 25$ is not solvable by radicals over the field of rationals.

Proof (F. Li 2006): using a Theorem by C. Bajaj 1988 and the factorization algorithm by E. R. Berlekamp 1967.

Cube-curve for this polynomial



detect $t_1, t_2 \in [0, 1]$ such that
polyline $P_0(t_0)P_1(t_1)P_2(t_2)P_3(t_3)$ is
fully contained in ρ .

Two corollaries from this example

Obviously and well-known:

Corollary 1: There is no exact arithmetic algorithm for calculating roots of polynomials (known since E. Galois; B.L. van der Waerden's example $p(x) = x^5 - x - 1$).

Corollary 2: There is no exact arithmetic algorithm for calculating ESPs (Chanderjit Bajaj, 1985; of order 20) in simple cube-curves (the new result; and order 6 only).

Note: not just a "rounding number problem" but a **FUNDAMENTAL** non-existence of exact algorithms, no matter what kind of time-complexity is allowed.

Approximate algorithms

An algorithm is a δ -approximation algorithm for a minimization problem P iff, for each input instance I of P , the algorithm delivers a solution that is at most δ times the optimum solution.

Example: General 3D ESP problem can be solved in $O(n^4 [b + \log(n/\varepsilon)]^2 / \varepsilon^2)$ time by an $(1 + \varepsilon)$ -approximation algorithm

(C. H. Papadimitriou 1985).

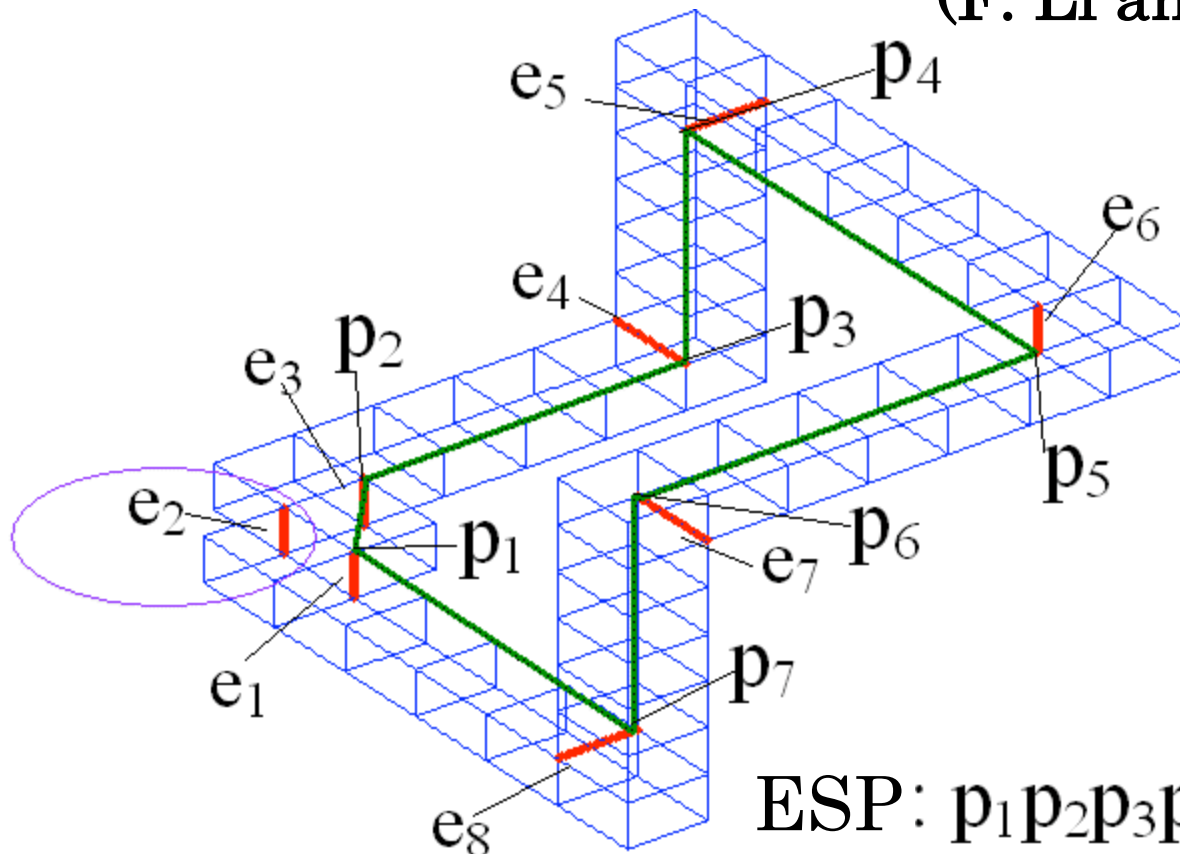
An algorithm is κ -linear iff its time complexity is in $\kappa(\varepsilon) \cdot O(n)$, and function κ does not depend on the problem size n , for $\varepsilon > 0$.

We use $\kappa(\varepsilon) = (L_0 - L) / \varepsilon$, where L is the true length of ESP, and L_0 the initial length.

First-class cube-curves

A cube-curve is *first-class* iff each critical edge contains one ESP vertex. Original RBA is correct and κ -linear, for first-class cube-curves.

(F. Li and R. Klette, CAIP 2005)

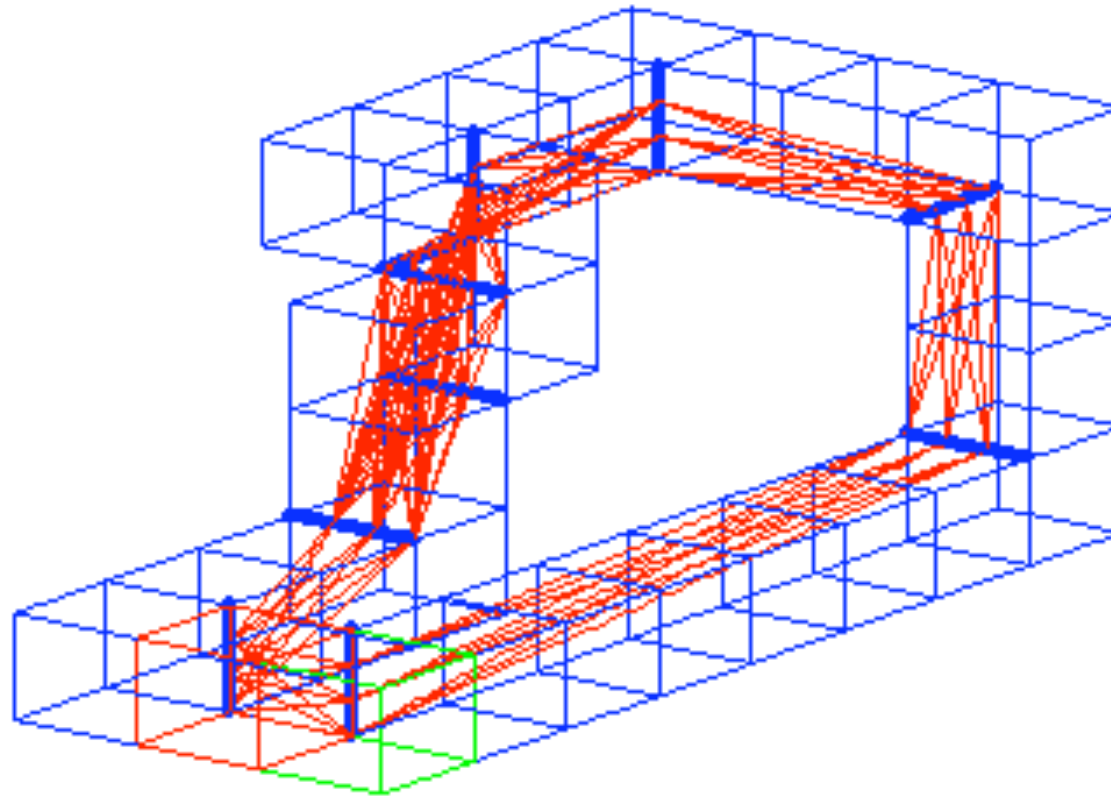


This cube-curve is not first-class.

ESP: $p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_1$

critical edges: $e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8$

Approximate graph-theoretical algorithm

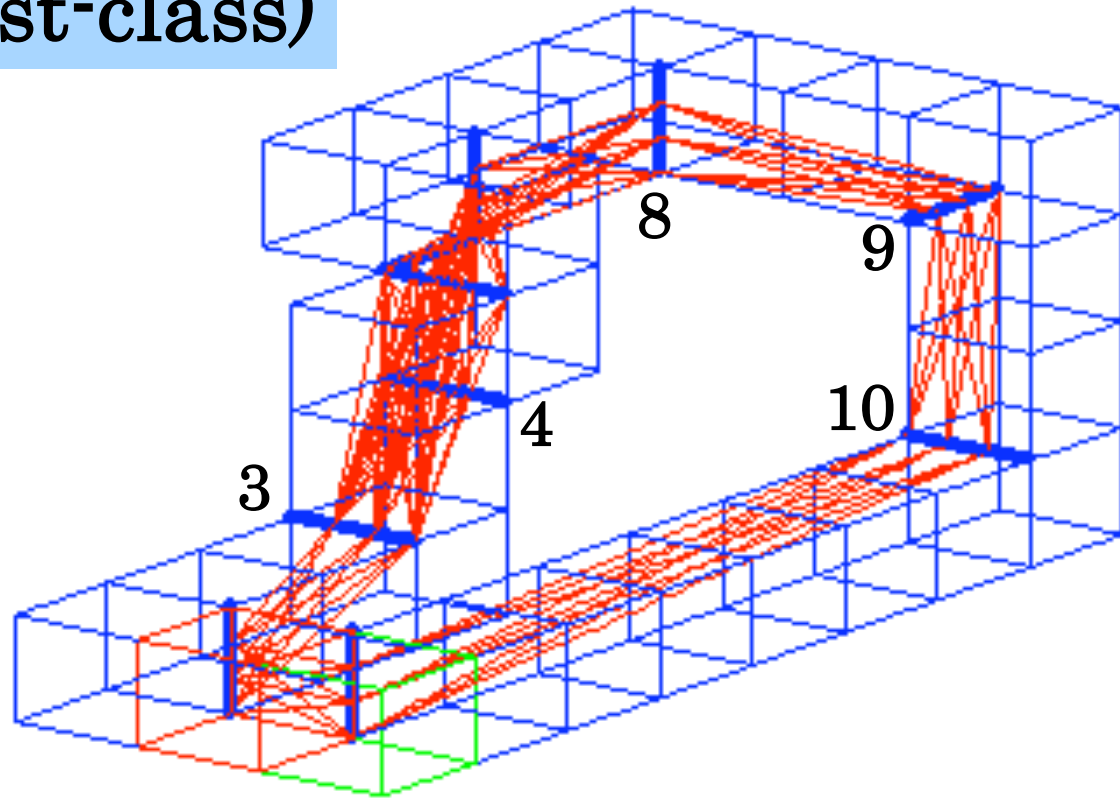


$$m = 3$$

Subdivide each critical edge by m uniformly-spaced vertices; connect each vertex with vertices such that resulting edge is contained in the tube. This defines a weighted undirected graph. Calculate a shortest-length cycle as (first-class !) input for original RBA.

Example (not first-class)

Red cube = first cube, containing critical edges 1 and 2. Yellow cube = last cube.



Subdivision number m	Edges with cycle vertices	Length of shortest cycle	Time (in seconds) in Matlab
3	{1, 2, 3, 6, 7, 8, 9, 10}	18.0252	1.2340

Time-complexity of graph-theoretic algorithm

$$O(m^4 n^4 + \kappa(\varepsilon) \cdot n)$$

This graph-theoretic algorithm applies Dijkstra's algorithm repeatedly; possibly its time-complexity can be reduced, but certainly not to be κ -linear.

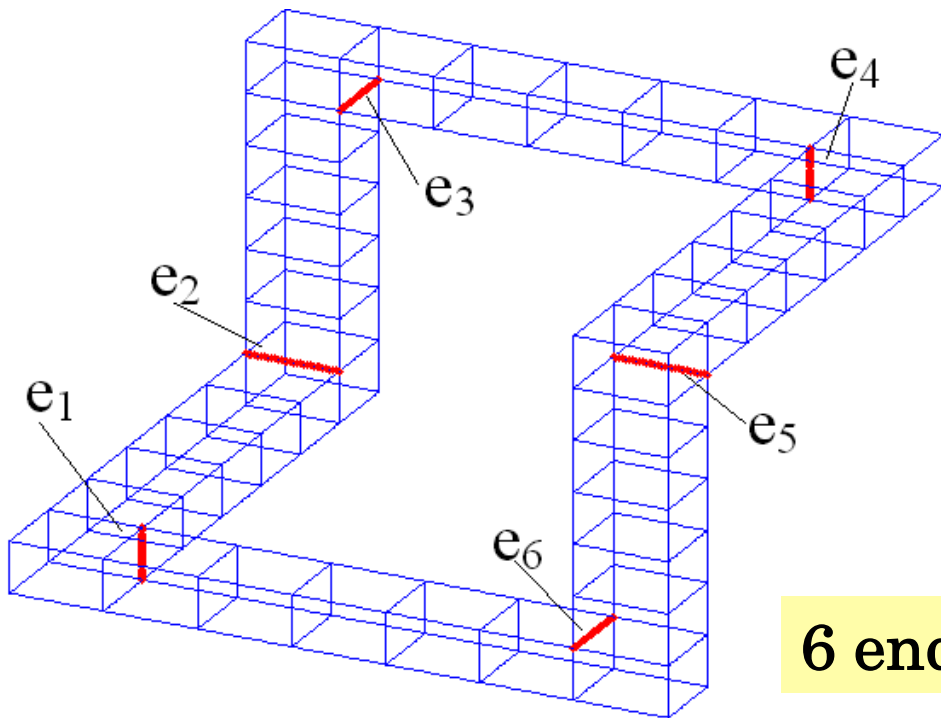
The benefit was in 2004 that this algorithm allowed for the first time to evaluate results obtained by the original RBA; all evaluations had been positive in those tests.

End angles and middle angles

Assume a simple cube-curve g and a triple of consecutive critical edges e_1 , e_2 , and e_3 such that e_i orthogonal to e_j , for $i, j = 1, 2, 3$ and $i \neq j$.

If e_1 and e_3 are also coplanar, then we say that e_1 , e_2 , and e_3 form an *end angle*, and a *middle angle* otherwise.

The ESP vertex on e_2 is an endpoint of e_2 .



6 end angles, no middle angle

Unique shortest paths

There is a uniquely defined shortest path, which passes through subsequent line segments $e_1, e_2, \dots,$ and e_k in 3D space in this order. - Three diff. proofs in

J. Choi, J. Sellen, and C.-K. Yap 1995

C.-K. Yap 1995

M. Sharir and A. Schorr, 1986

Obviously: vertices of a shortest path can be at real division points, and even at those which cannot be represented by radicals over the field of rationals

Approximate numerical algorithm

We assume that the given cube-curve is first-class and has at least one end angle where we split the curve into arc(s).

For each arc, one vertex on each critical edge can be calculated using optimization (differentiation) of 2- or 3-variable equations. A variable t_i determines the position of vertex p_i on edge e_i .

Experiments showed again that the original RBA and this (efficient) approximate algorithm lead to (basically) identical results; which was of benefit for evaluations.

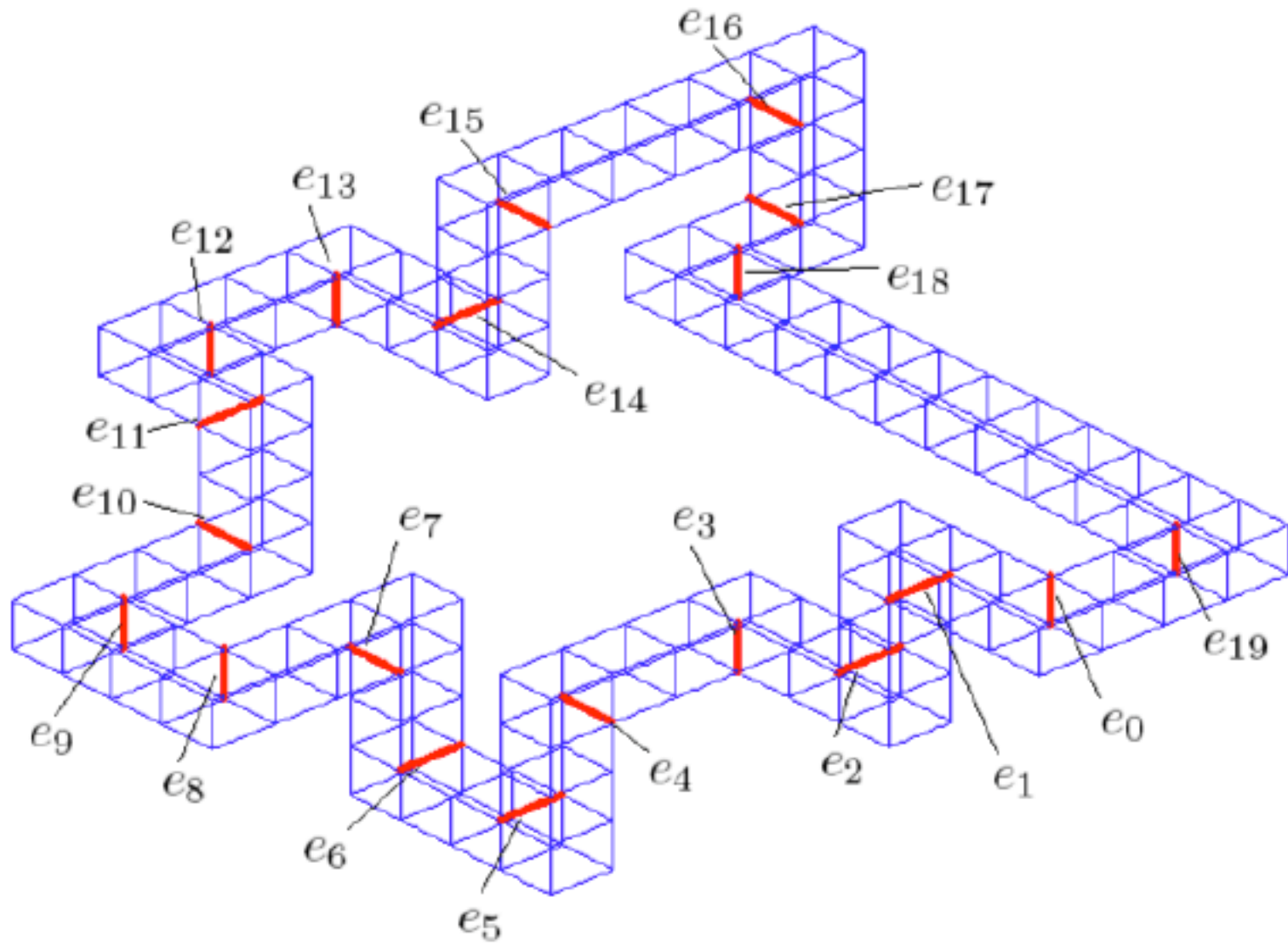
An open problem in 2003 and its 2005 answer

An open problem (see [KletteRosenfeld2004, page 406]) was stated as follows: Is there a simple cube-curve such that none of the nodes of its ESP is a grid vertex?

The answer is “yes”, and any of those does not have any end angle! Thus, the provably correct approximate numerical algorithm cannot be used in general.

This lead us back to the original RBA:

- is it correct? (use either approximate graph-theoretical or numerical algorithm for evaluation)
- what is its time-complexity in general?



A simple cube-curve where the ESP does not have any grid-point vertex (and which has no end angle).

Two open problems

What is the smallest (say, in number of cubes or in number of critical edges - both is equivalent) simple cube curve which does not have any end angle.

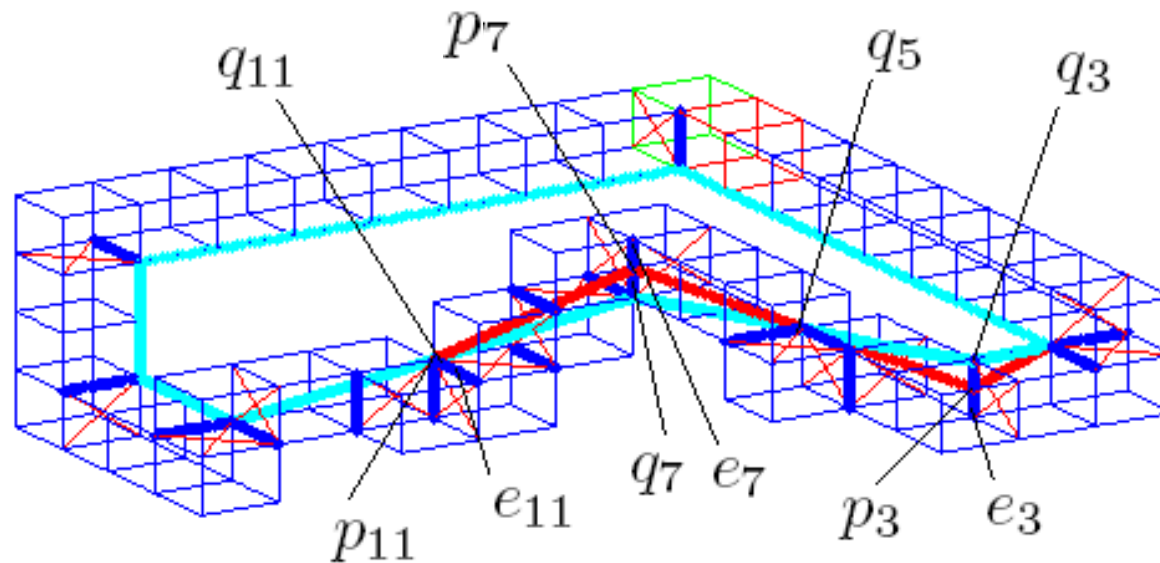
What is the smallest (say, in number of cubes or in number of critical edges - both is equivalent) simple cube curve which does not have any of its MLP vertices at a grid point location?

We assume, the second problem is more difficult to solve.

Necessary corrections in the original RBA

- OP_2 : if intersecting with the triangle $p_{i-1}p_i p_{i+1}$ and using the convex arc only, we may miss edges of the step set - more tests are needed, and this option was reformulated (a complex correction).
- OP_3 : the vertex p_{new} , found by optimization, may specify edges $p_{i-1}p_{\text{new}}$ and $p_{\text{new}}p_{i+1}$ such that one or both of them are not fully contained in the tube of the curve; an additional test is needed (a simple correction).

A Situation Where the Original Option 2 Fails



The cyan polyline $q_3q_5q_7q_{11}$ is shorter than the red polyline $p_3p_7p_{11}$. However, according to Option 2, $p_3p_7p_{11}$ can not be improved. In other words, limited by (the original) Option 2, we can not use the shorter polyline $q_3q_5q_7q_{11}$ instead of polyline $p_3p_7p_{11}$. This is because the vertices q_3 and q_7 are not in the set of the intersection points between any critical edge in the set $\{e_i : i = 4, 5, 6, 7, 8, 9, 10\}$ and the closed triangular region $\triangle(p_3, p_7, p_{11})$.

Final result for ESPs in simple cube-curves

Corrections define the *edge-based corrected RBA*.

Instead of moving points along critical edges, we can also move points within *critical faces* (which contain one critical edge). This defines the (slower) *face-based corrected RBA*.

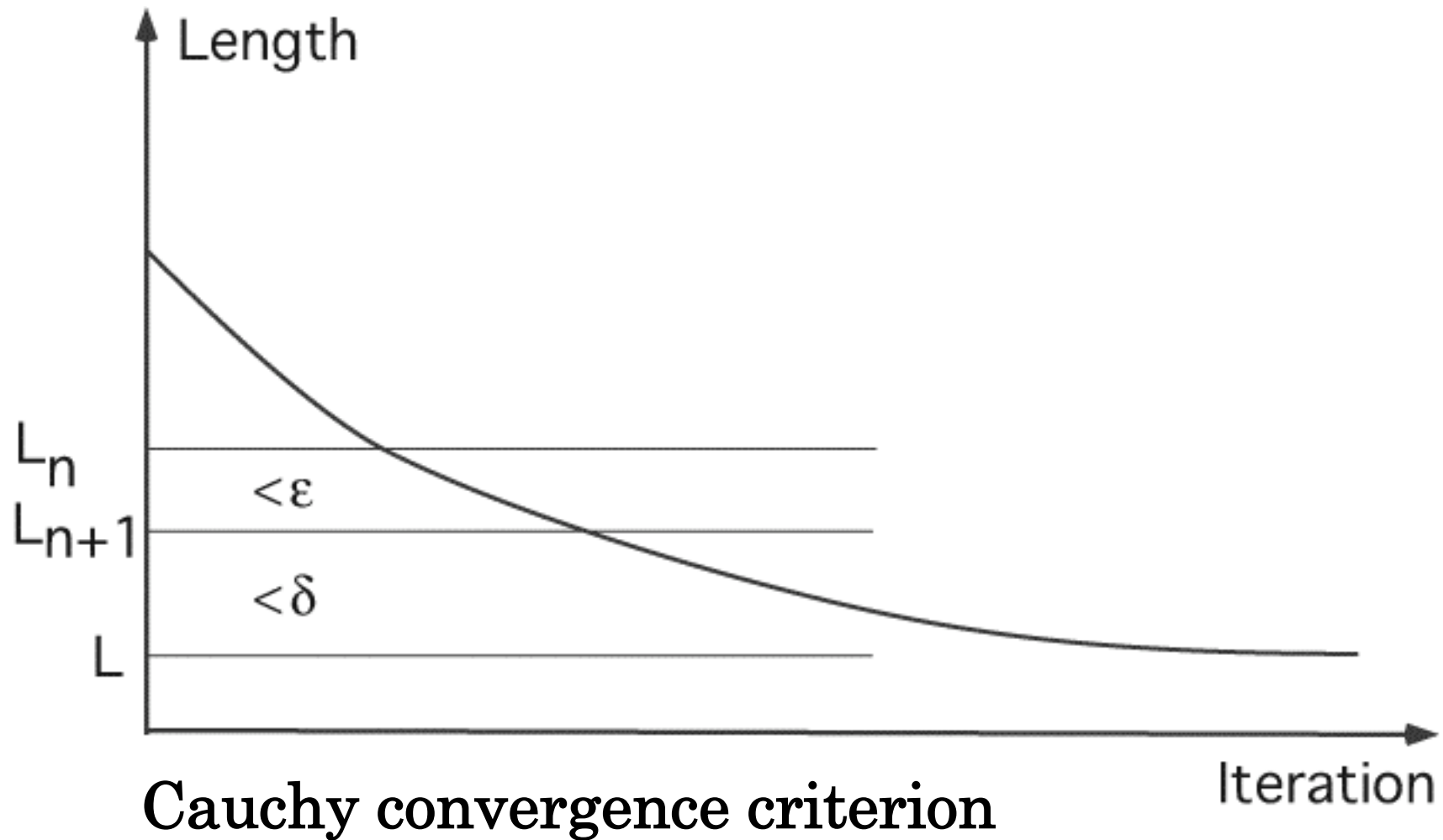
Both are provably correct, implemented in Java, and have provably asymptotic time complexity

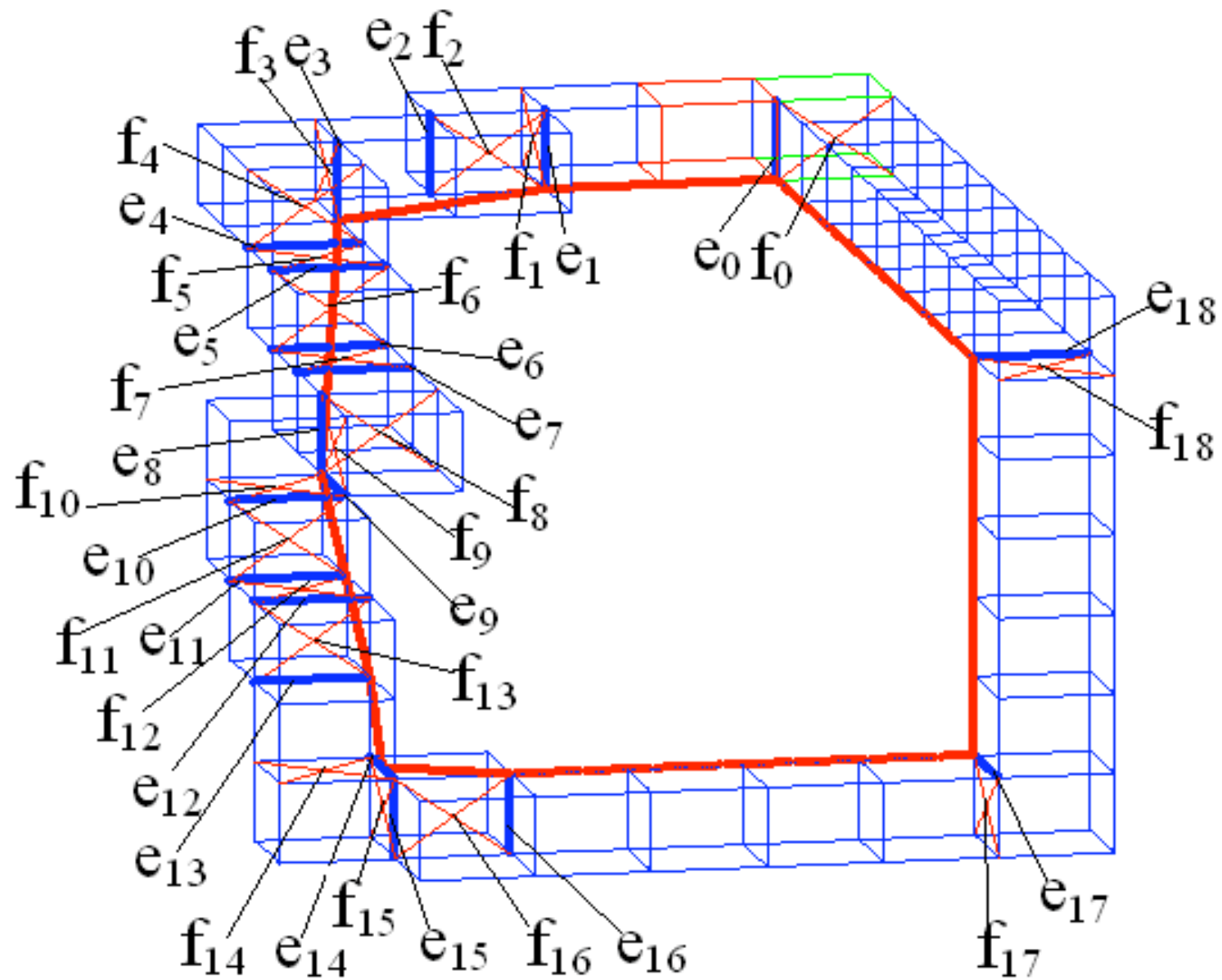
$$\kappa(\varepsilon) \cdot O(n)$$

(i.e., they are κ -linear).

Recall: $\kappa(\varepsilon) = (L_0 - L) / \varepsilon$

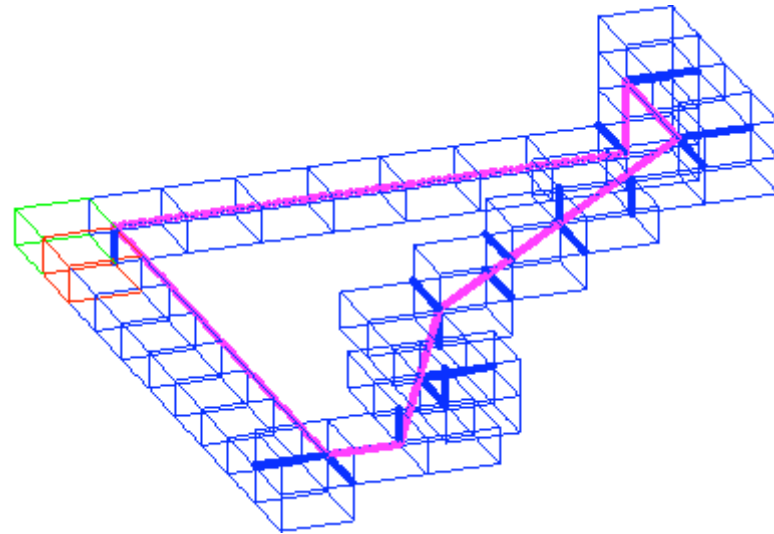
Let ε be the maximum accuracy of your program. We can obtain arbitrary accuracy (with respect to L) when continuing iterations.



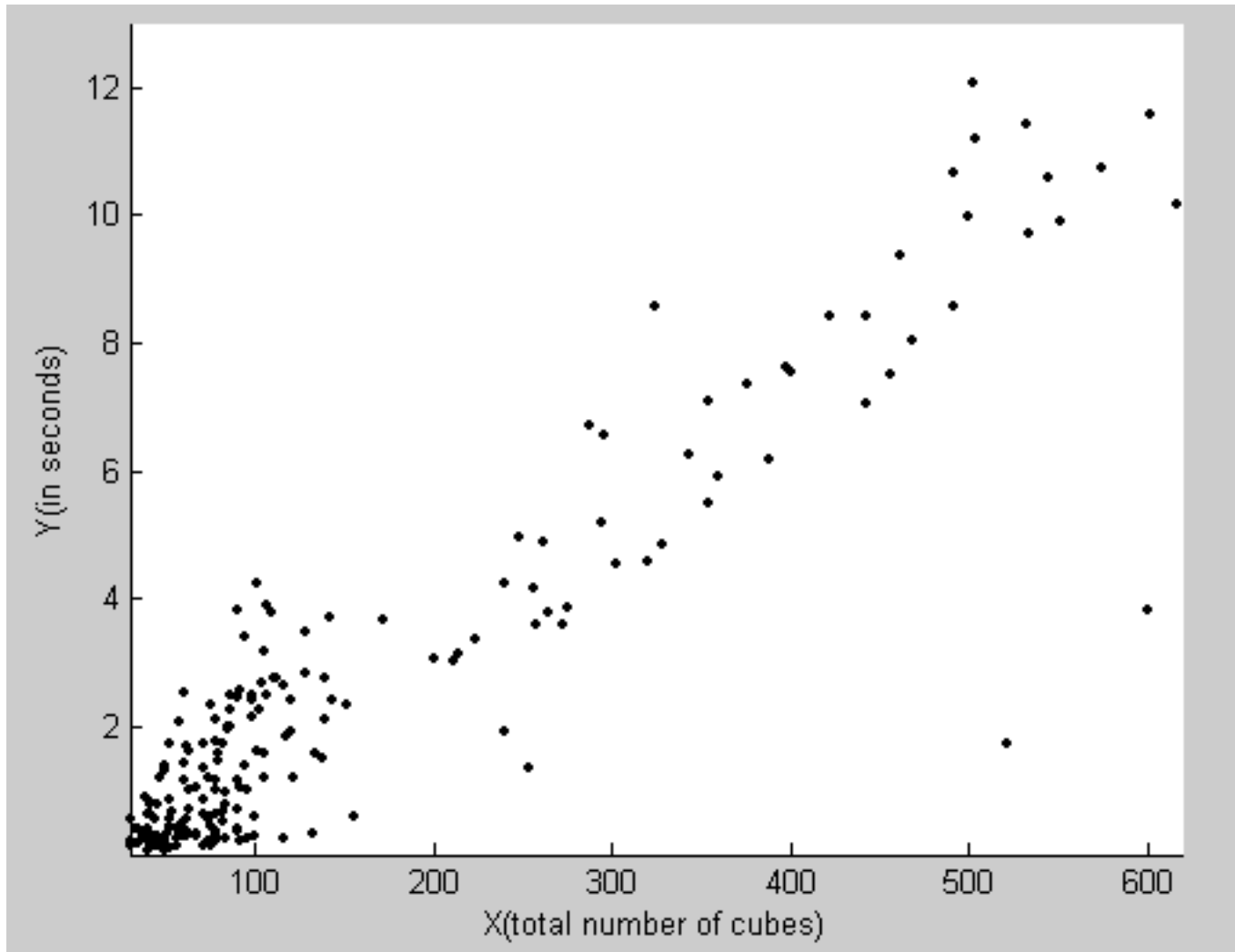


Random generation of simple cube-curves is an interesting problem on itself.

Animation: RBA in Action (27 iterations)



Time for RBA on Simple Cube Curves $\varepsilon = 10^{-10}$



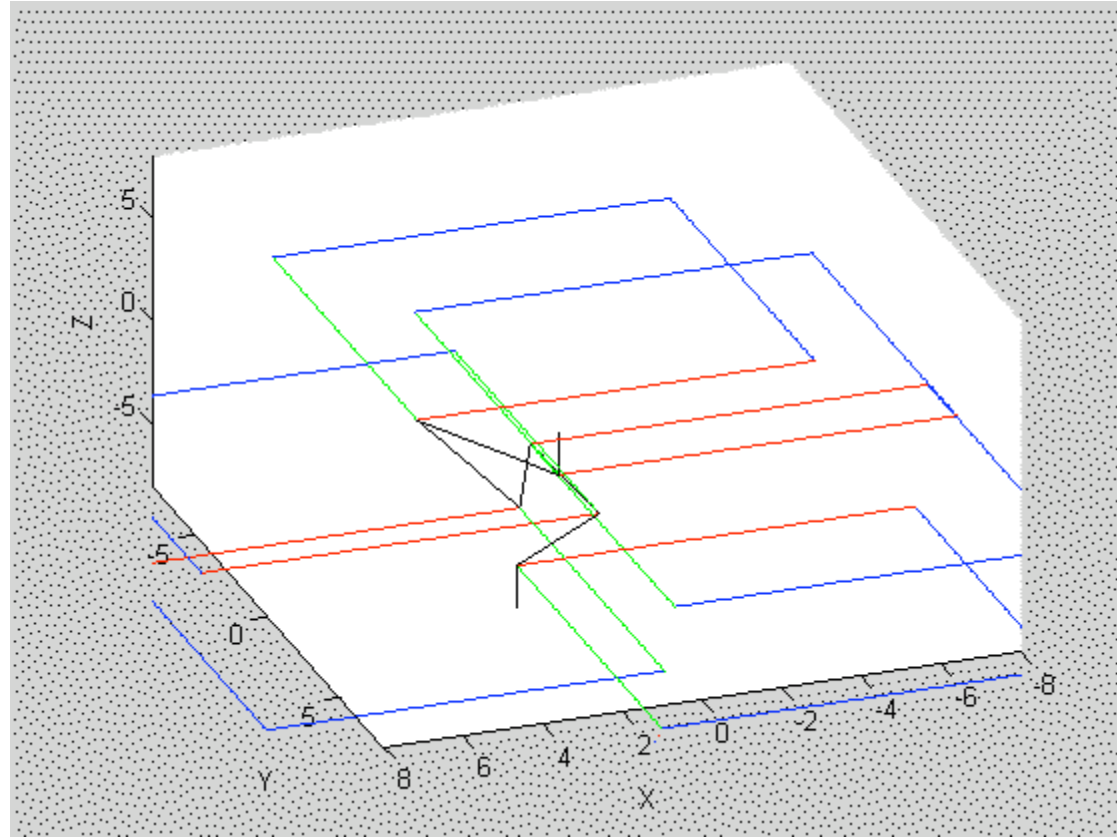
Implemented in Java, run under Matlab 7.0.4, Pentium 4

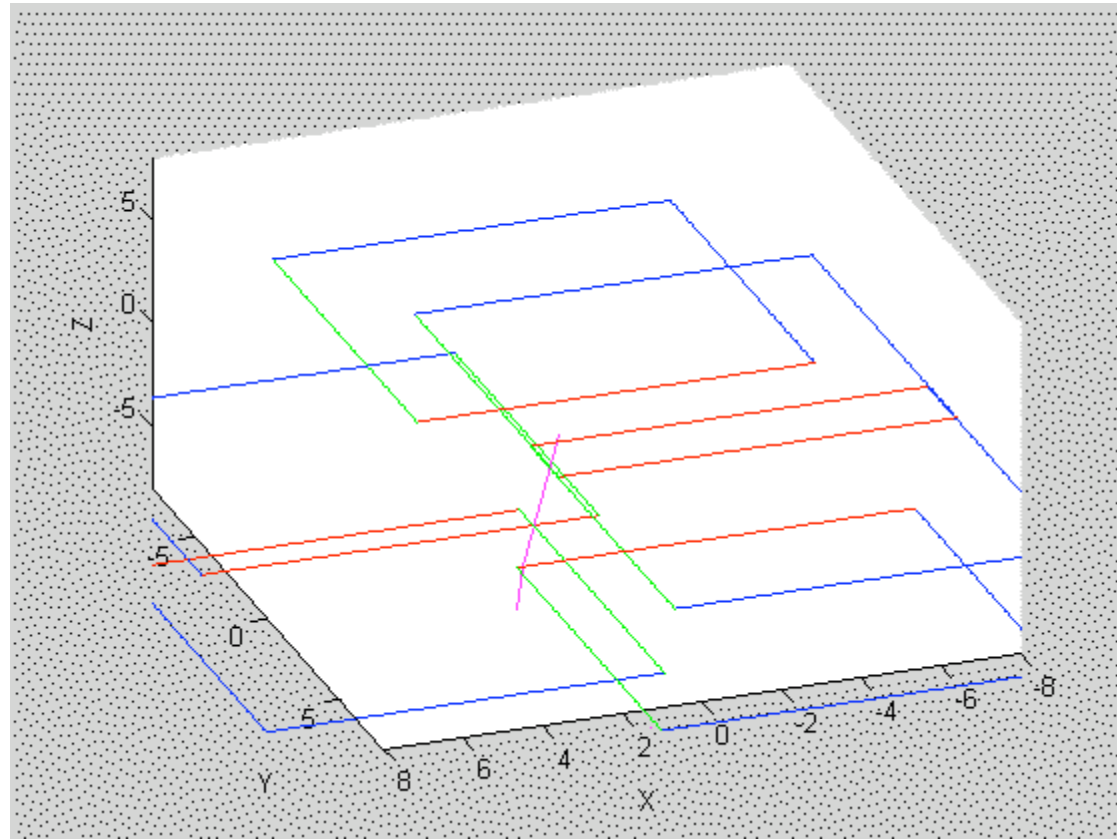
RBA applications to further ESP problems

In this talk, we discuss only two examples. For more applications in computational geometry, see report 2141 by F. Li and R. Klette on IMA's Website (in Minneapolis).

Example 1: finite stack of q - rectangles defining the (convex) obstacles (NP-complete, J.S.B. Mitchell and M.Sharir, 2004)

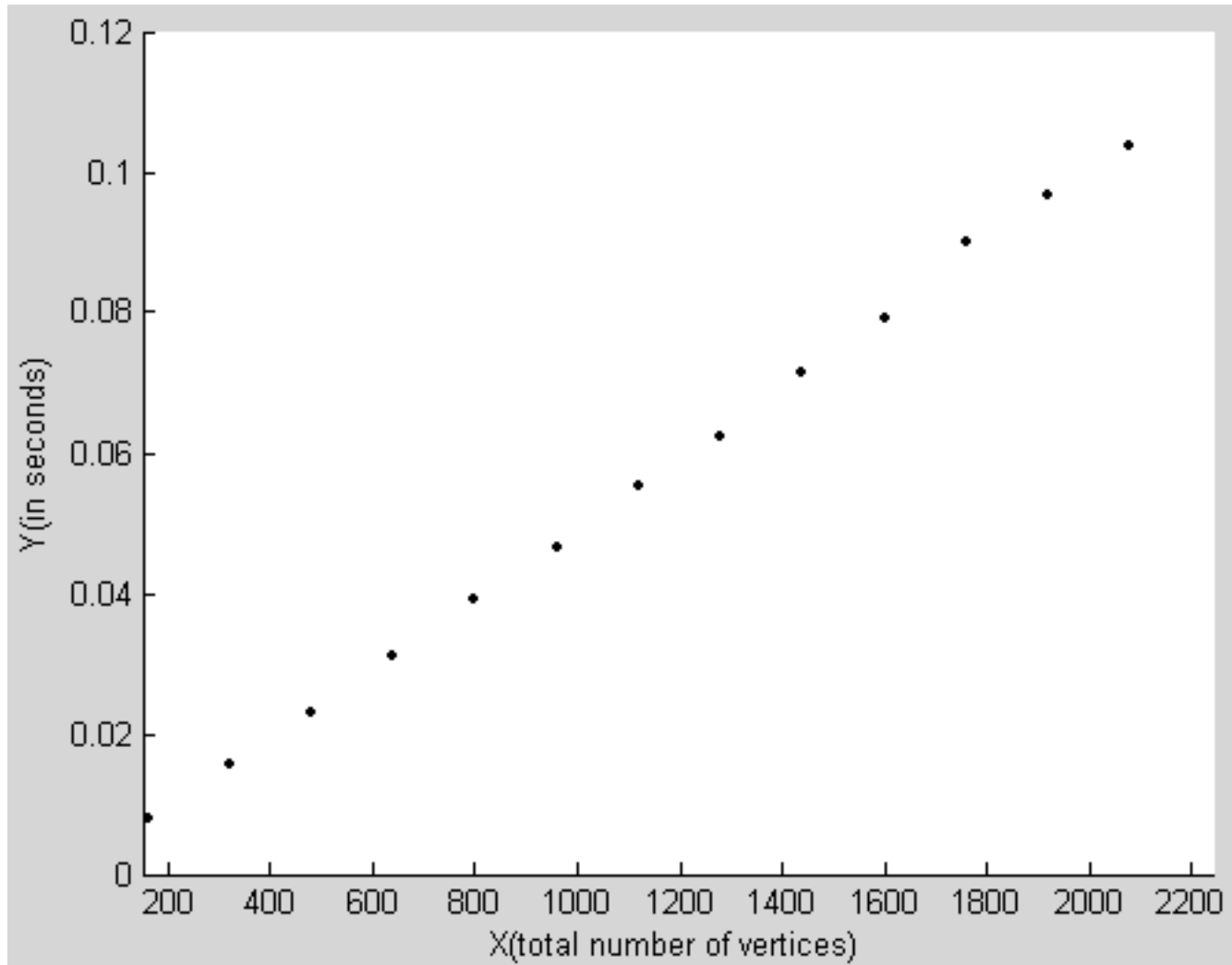
Example 2: touring a finite sequence of simple (not necessarily convex) polygons (open problem, M. Dror, A. Efrat, A. Lubiw, and J.S.B. Mitchell, 2003)





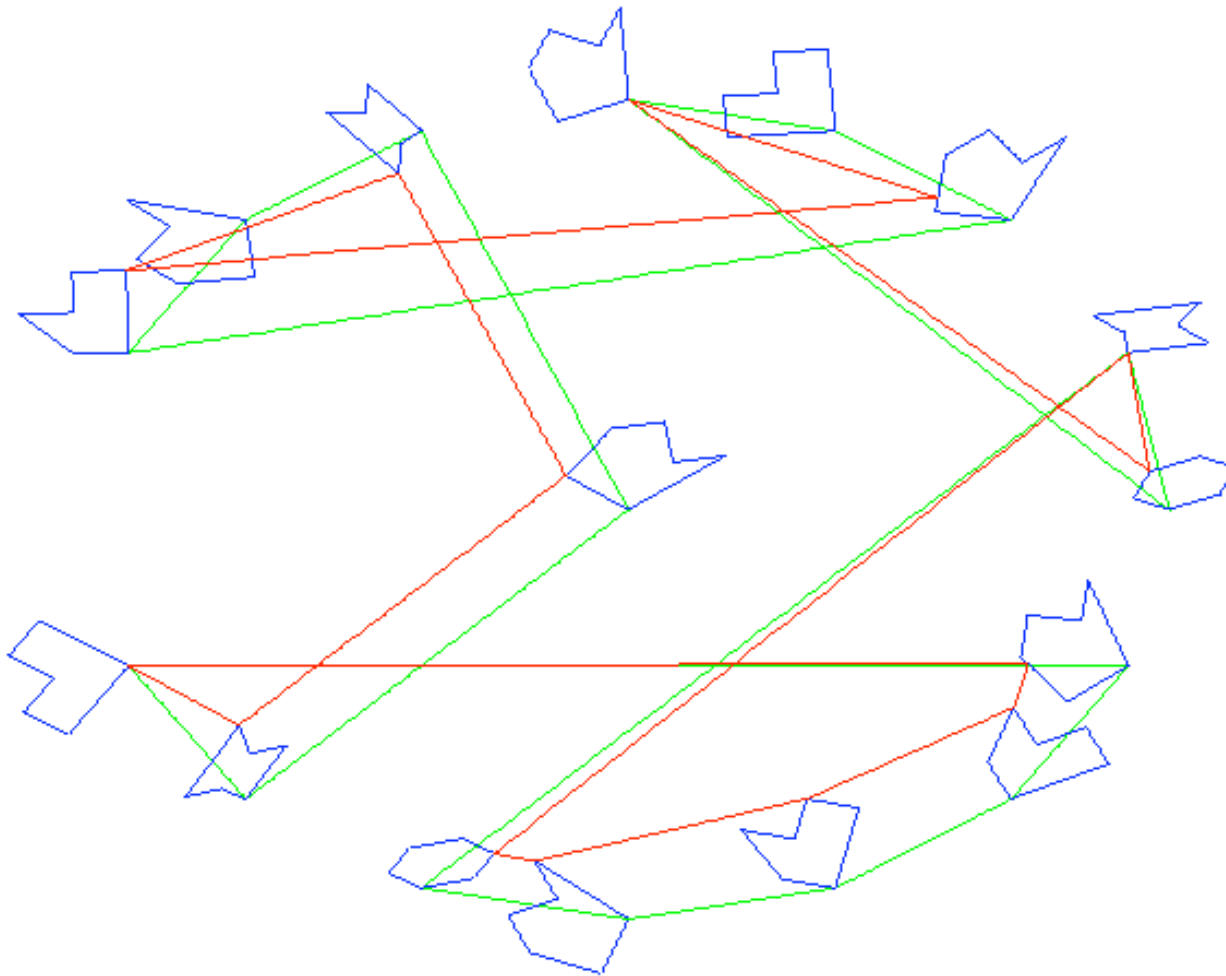
Time for RBA on q-Rectangles

$$\varepsilon = 10^{-10}$$

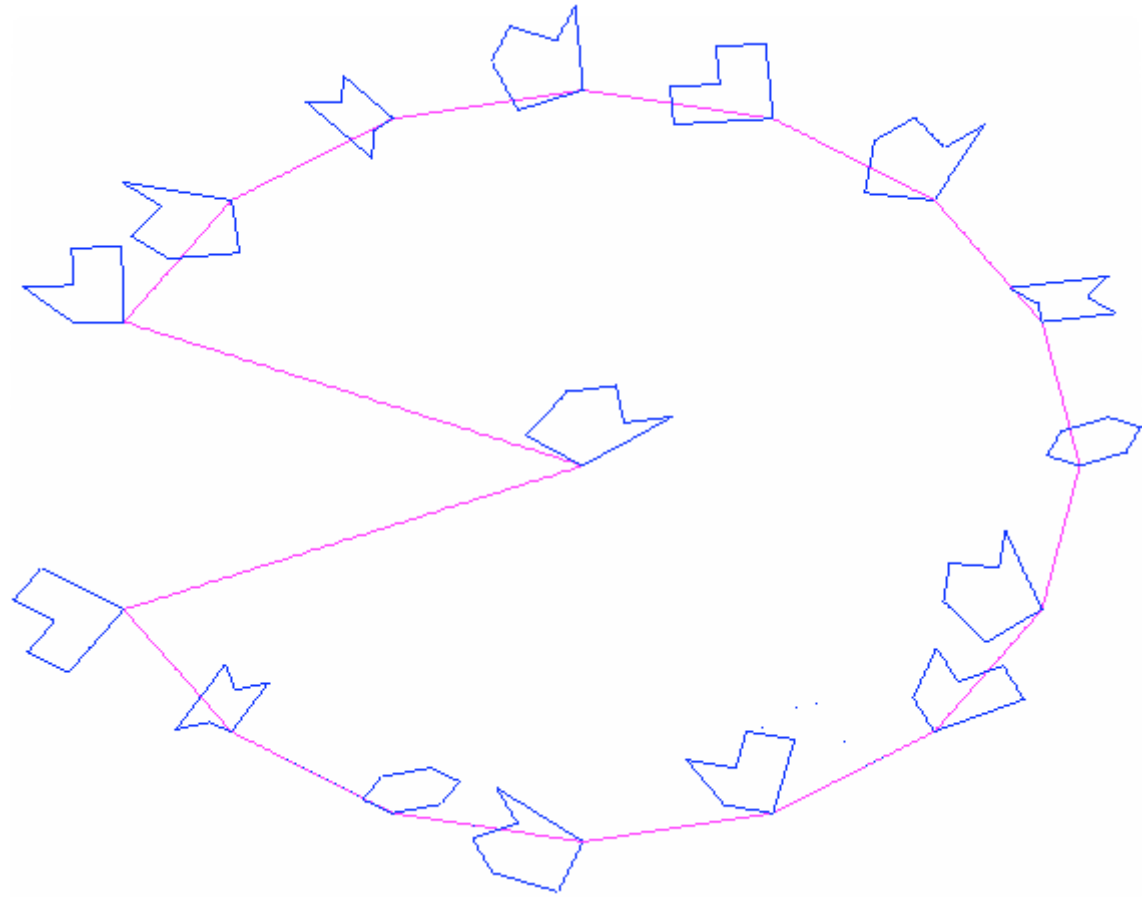


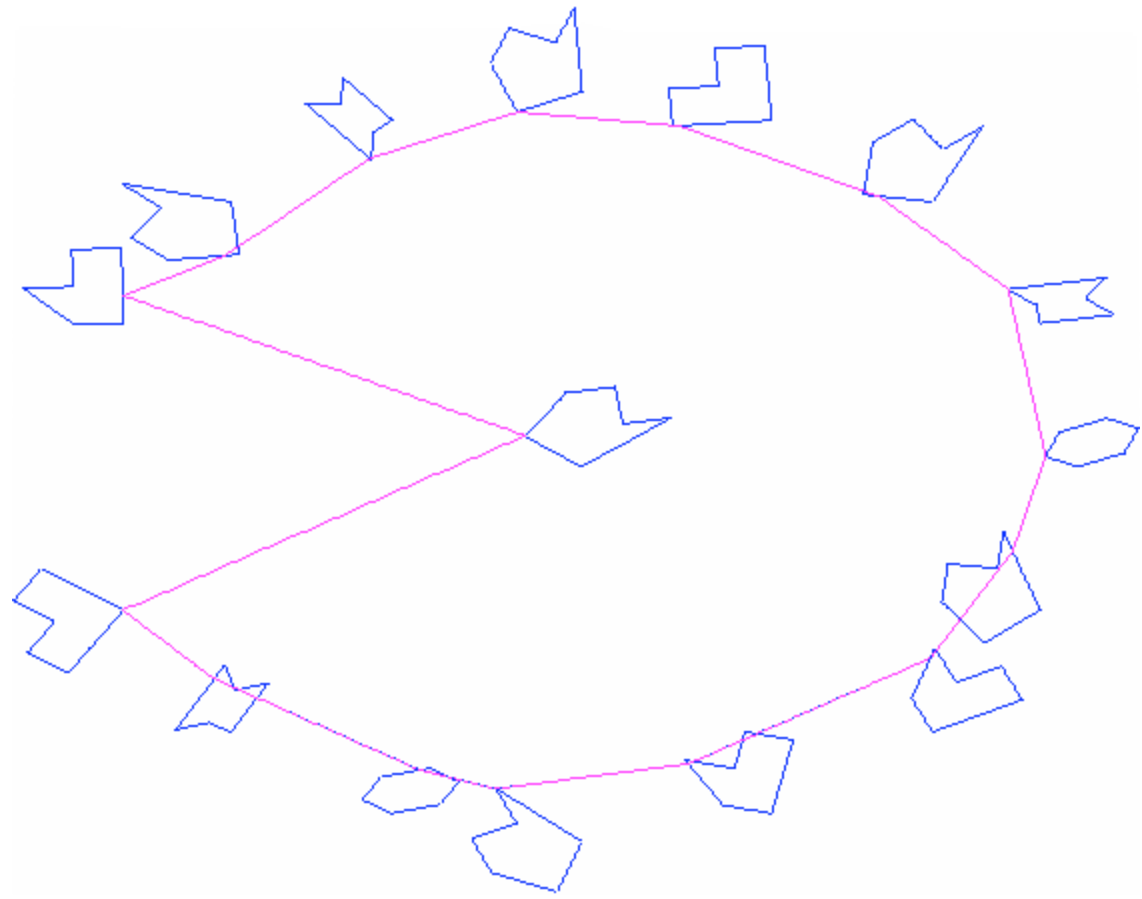
Implemented in Java, run under Matlab 7.0.4, Pentium 4

Touring a Finite Sequence of Polygons

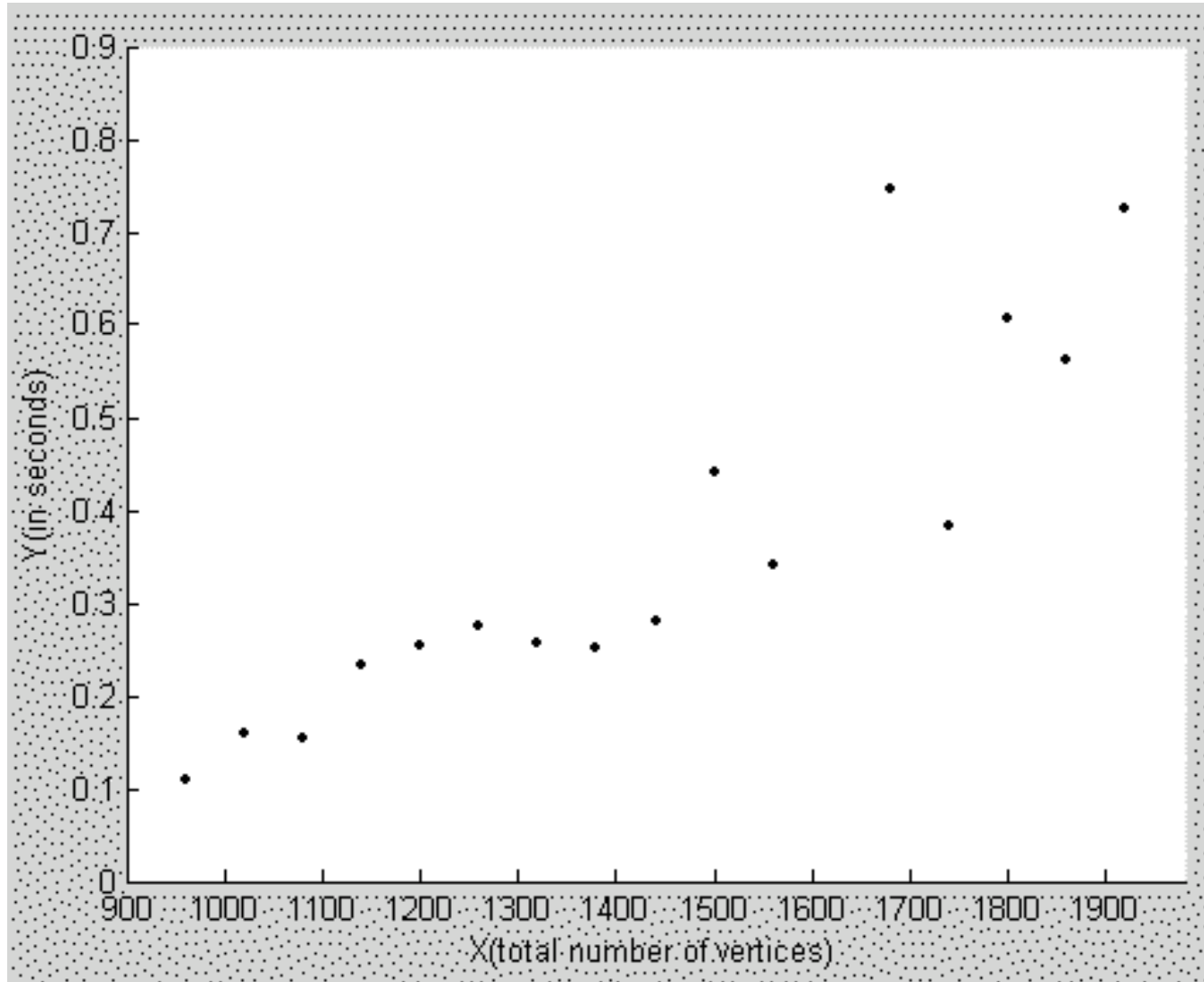


Note: was until now an open problem for non-convex polygons





Time for RBA on Sequence of Polygons $\varepsilon = 10^{-10}$



Implemented in Java, run under Matlab 7.0.4, Pentium 4

Conclusions

Idea of the rubberband algorithm was generalized to establish a whole class of RBAs:

- assume an ESP problem
- define and select (or calculate) the step set
- apply the RBA, that means:
 - basic idea is Option 3
 - define ε to be the maximum numeric accuracy you can achieve on your machine for a selected numerical precision
 - repeat local optimization until no further improvement is possible
 - if step set correct, then local = global minimum