Intelligent and Adaptable Software Systems

Advanced Algorithms: Optimization and Search Methods

Dr. Arno Formella

Computer Science Department
University of Vigo

12/13

Course organization

course notes

- Homepage:

`http://trevinca.ei.uvigo.es/~formella/doc/ssia12`

- whiteboard
  (illustrations, notations, ideas for proofs, algorithms etc.)
- very short introduction to certain aspects related to optimization and search methods, and some applications

Course organization

class room hours (preliminary)

Optimization and Search Methods
fridays, 16:00–18:00

28.09. class	05.10. class	19.10. (no-class)	26.10. class	02.11. lab
09.11. class	16.11. class	23.11. class	30.11. lab	07.12. lab
14.12. class	21.12. class	11.01. class	18.01. class	25.01. eval

Course organization

class room hours

- Dr. Arno Formella
  office hours: tuesdays, 09:30-13:30 and 17-19

Bibliography

books

- OUR 519.8.15, OUR 519.8/23, OUR 519.8/24, OUR 519.8/46, OUR 519/17, OUR 519/20

Your work

homework, lab hours, presentations

- browse through the web pages provided in the following slides
- sort the information provided into the categories of optimization methods as mentioned below
- find a web service that allows you to compute the derivation of a function
- use the NEOS–server to find the minimum of the function

$$f(x) = a(x-b)^2 + c + d\cos(e(x-f)+g)$$

for some (different) values of the parameters (maybe you start with $d = e = f = g = 0$).

Your work

more extensive research task I

1. form a group with at most one other student
2. select in accordance with Prof. Arno Formella one of the proposed algorithms on the next slide
3. elaborate a not too short and not too long article (6 to 10 pages) about the algorithm, including at least the aspects stated on the next but one slide.

Your work

more extensive research task II, examples

- Nelder Mead algorithm
- Newton Raphson
- Rodríquez García-Palomares algorithm
- Levenberg Marquardt algorithm
- great deluge algorithm
- local unimodel sampling

Your work

more extensive research task III

your article should treat the following issues

- description of the algorithm
- main field of application
- advantages and disadvantages compared to other algorithms
- available software/implementations
- critical discussion of their APIs
- references on the algorithm and its applications

Bibliography I

links

(working in september 2012)

- `http://www.neos-server.org`
  online optimization project
- `http://www.coin-or.org/index.html`
  pperation research
- `http://www.cs.sandia.gov/opt/survey`
  global optimization
- `http://www.mat.univie.ac.at/~neum/glopt.html`
  global optimization

Bibliography II

links

- `http://www.stanford.edu/~boyd/index.html`
  Stephen P. Boyd, Stanford
- `http://iridia.ulb.ac.be/~mdorigo/ACO/`
  ant colony optimization
- `http://plato.asu.edu/gom.html`
  continuous global optimization software
- `http://www.swarmintelligence.org/index.php`
  particle swarm optimization
- Rui Mendes. *Population topologies and their influence in particle swarm performance*. PhD Thesis, Universidad de Minho, 2004.
  `http://www.di.uminho.pt/~rcm/`

Motivation

what is it?

Optimizing means

- search for (at least) one solution
- which is different from other possible solutions
- in the sense of being (sufficiently) extreme
- within an ordering
- possibly taking into account certain restrictions
- (within a certain limit of computing time).

Example: hiking in a mountain ridge (with fog).

Motivation

examples

Problems which one wants to solve:

- minimizing cost
- maximizing earnings
- maximizing occupation
- minimizing energy
- minimizing resources

Basic concepts

observations

the search space and/or the objective function can be

- discrete or continous
- total or partial
- simple or complex, especially in respect to evaluation time
- explicite, implicite, experimental
- linear or non-linear
- convex or non-convex
- differentiable or non–differentiable
- constrained or unconstrained
- static or dynamic

The objective function must be confined.

Basic concepts

objective functions

- Minimization
- Maximization
- Obviously any maximization problem can be converted to a minimization problem.

Basic concepts

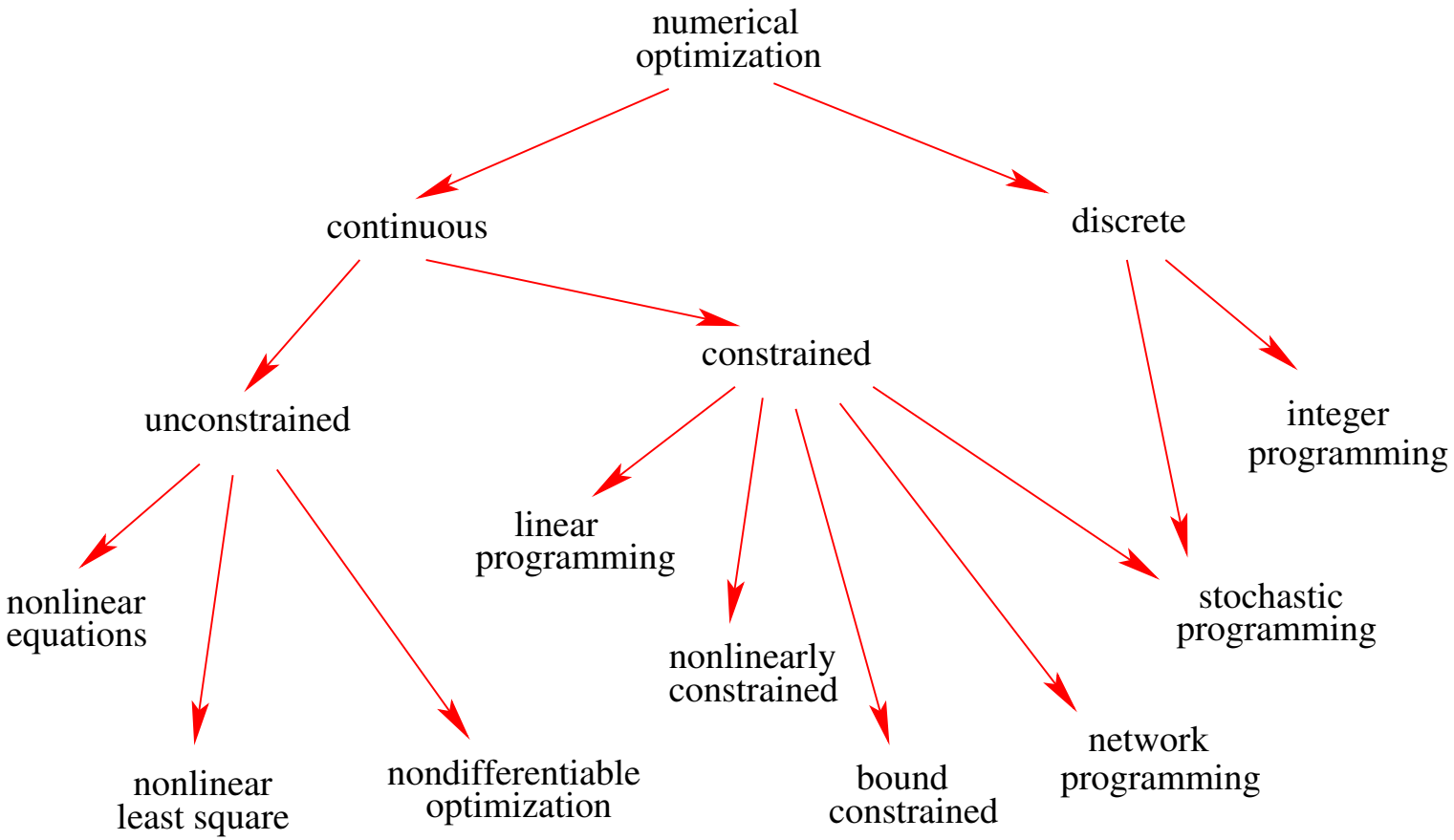
conditions

- restrictions
- feasable solution (feasibility problem)
- coding of the solutions

Basic concepts

classification

(after NEOS server (almost), Argonne National Laboratory)



Basic concepts

types

to be distinguished

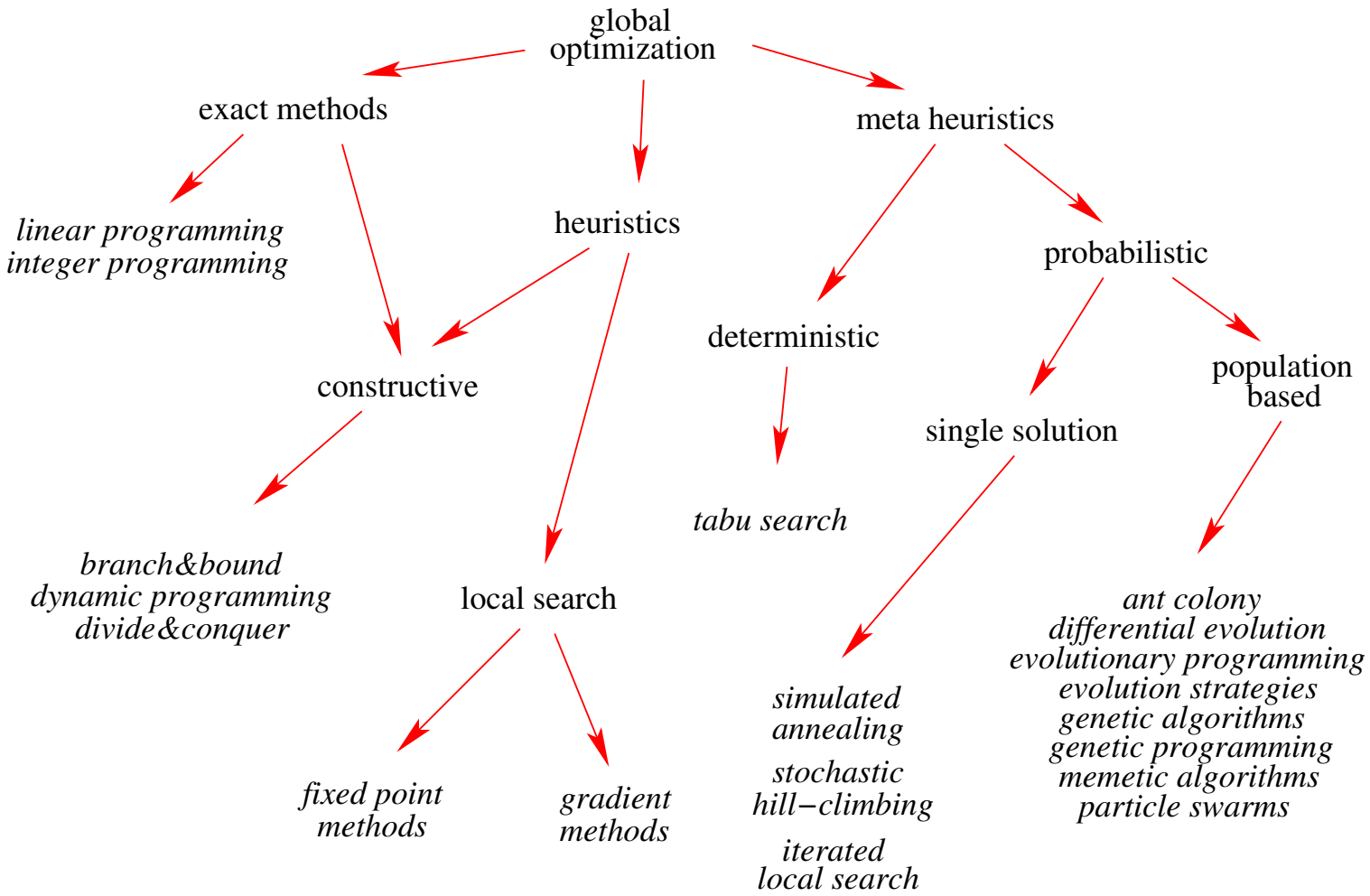
local optimization: usually one starts from an initial solution and stops when having found a local (close) minimum

global optimization: one tries to find the best solution globally (among all possible solutions)

- The main problem of global optimization is: getting trapped in a local minimum (premature convergence)

Basic concepts

global optimization (incomplete intent)



A real application

psm

approximate Point Set Match in 2D and 3D

An application where we need sophisticated search and optimization techniques.

Dónde está Wally?

Motivation

psm

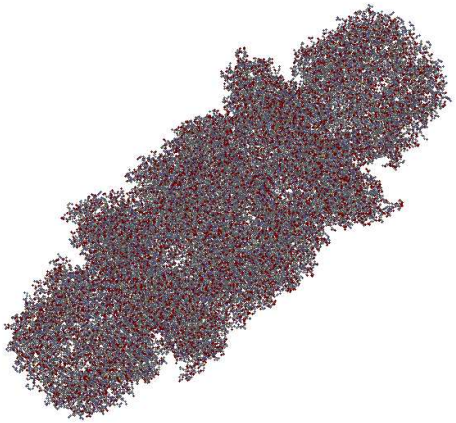
- searching of patterns
  (relatively small sets of two– or three–dimensional points),
  within search spaces
  (relatively large point sets)
- comparing point sets
- key words
  geometric pattern matching, *structure comparison*, *point set matching*, *structural alignment*, *object recognition*

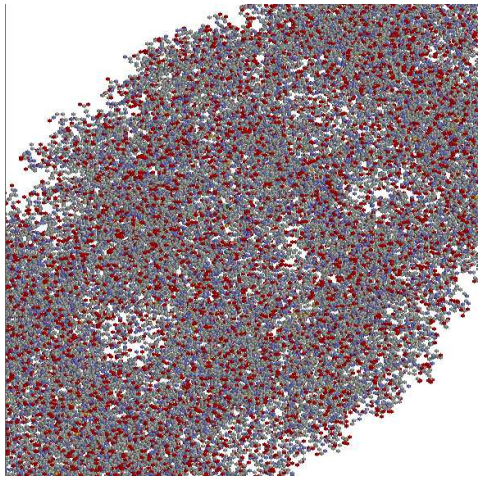
Joint work with

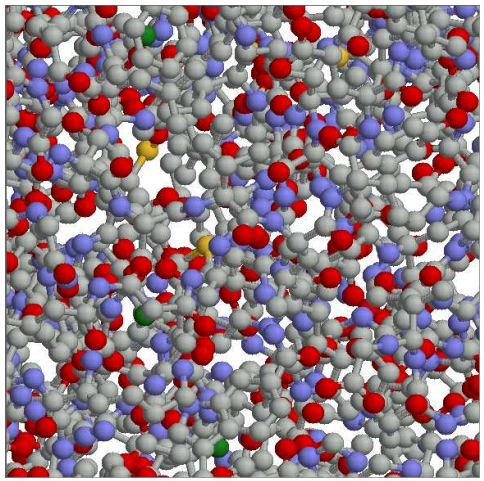
- Thorsten Pöschel
- some ideas from: Kristian Rother, Stefan Günther
- Humboldt Universität—Charité Berlin
  `http://www.charite.de/bioinf/people.html`
- `psm` is one of the algorithms available at
  `http://farnsworth.charite.de/superimpose-web`

Search of a substructure in a protein

search space

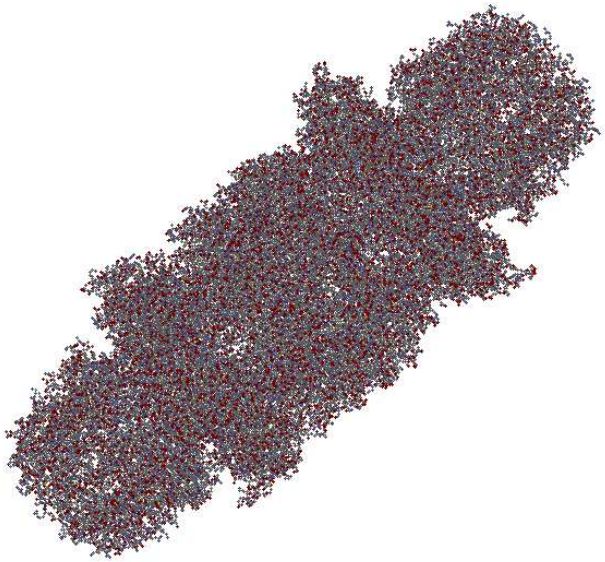


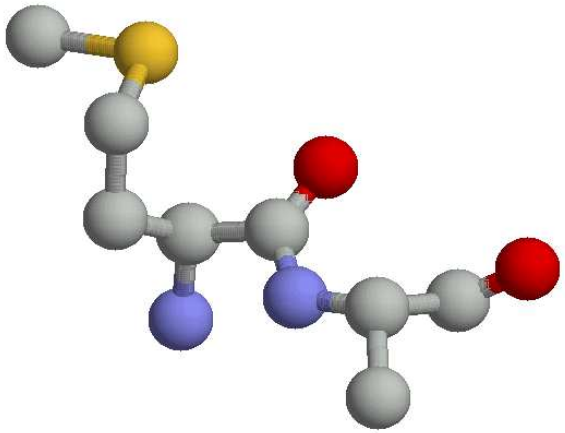




Search of a substructure in a protein

search pattern





Dónde está Wally?

Informal problem description

- given a search space and
- a search pattern,
- find the location within the space which represents best the pattern

Extensions

- find the best part of the pattern
  which can be represented within the search space
- allow certain types of deformation of the pattern
- find similar parts within the same point set

Formal problem description

- search space:
  $S = \{s_0, s_1, \ldots, s_{n-1}\} \subset \mathbb{R}^d, \qquad |S| = n$
- search pattern:
  $P = \{p_0, p_1, \ldots, p_{k-1}\} \subset \mathbb{R}^d, \qquad |P| = k \leq n$
- dimension $d = 2$ or $d = 3$

Search and alignment

- the aligning process can be separated in two parts
  - find the matching points in the pattern and the search space
  - find the necessary transformation to *move* the pattern to its location
- an approximate alignment must be qualified

Matching

- a matching is a function that assigns to each point of the search pattern a different point of the search space
- $\mu : P \longrightarrow S$ injective, i.e.,
- if $p_i \neq p_j$ then $\mu(p_i) \neq \mu(p_j)$
- let's write: $\mu(p_i) = s'_i$ and $\mu(P) = S'$

Transformations

- transformations which maintain distances:
  translation, rotation and reflection
- transformations which maintain angles:
  translation, rotation, reflection and scaling
- deforming transformations:
  shearing, projection, and others (local deformations)

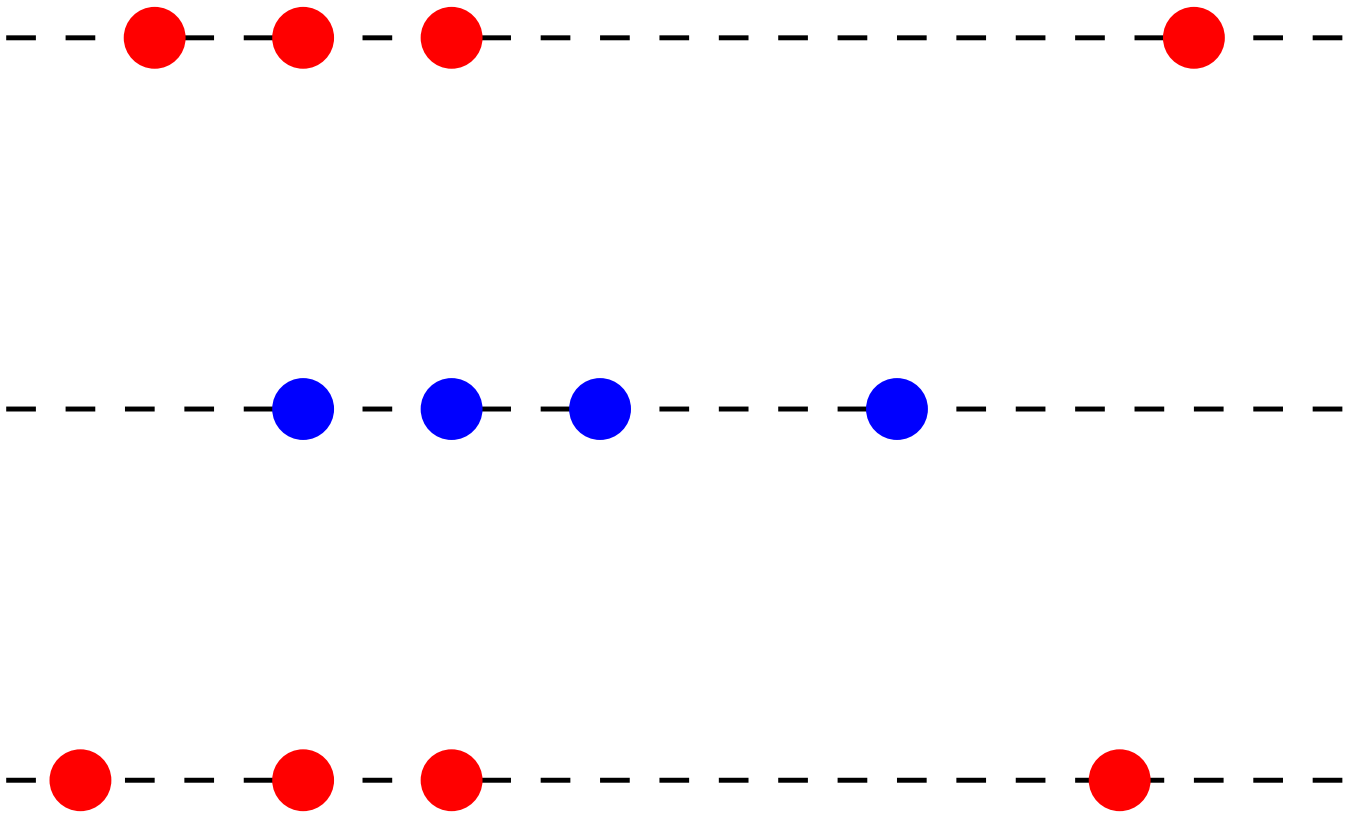
Congruent and similar transformations

- rigid motion transformation
  (euclidean transformation or congruent transformation)
  only translation and rotation
- similar transformation
  rigid motion transformation with scaling
- we may allow reflections as well (L–matches)
- let T be a transformation (normally congruent)
- we transform the pattern
- let's write: $T(p_i) = p_i'$ and $T(P) = P'$

Alignment

- a matching μ together with a transformation T is an alignment (μ, T)
- rigid motion transformation: congruent alignment
- with scaling: similar alignment
- with reflection: L–alignment

One–dimensional example



Distance of an alignment

- let (μ, T) be an alignment of P in S
- we can measure the distances between transformed points of the pattern and their partners in the search space
- i.e., the distances

$$d_i = d(T(p_i), \mu(p_i)) = d(p'_i, s'_i)$$

- obviously, if $d_i = 0$ for all i
  then the alignment is perfect

Examples of different distances of an alignment

- root mean square distance (RMS)

$$d = \sqrt{\frac{1}{n}\sum_i (p'_i - s'_i)^2}$$

- average distance (AVG)

$$d = \frac{1}{n}\sum_i |p'_i - s'_i|$$

- maximum distance (MAX)

$$d = \max_i |p'_i - s'_i|$$

Quality of an alignment

- there are many interesting distance measures
- a distance d has its value in $[0, \infty[$
- we use the quality Q of an alignment $Q = 1/(1+d)$
- (other possibility: $Q = \exp(-d)$)
- hence: $Q = 1$ perfect alignment, $\quad Q \in]0, 1]$
- and: $Q < 1$ approximate alignment

Formal problem description

- given a search space S, and
- given a search pattern P
- given a distance measure
- find an alignment (μ, T) of P in S
  with minimum distance d (or maximum quality Q)

Perfect congruent alignments

- congruent alignments in $\mathbb{R}^3$ (Boxer 1999):

$$O(n^{2,5}\sqrt[4]{\log^* n} + \underbrace{kn^{1,8}(\log^* n)^{O(1)}}_{\text{output}} \log n)$$

- for small k the first term is dominant
- $\log^* n$ is smallest l such that

$$2^{2^{\cdots^2}} \Big\} l-\text{veces} \geq n \qquad \log^* n = 5 \implies n \approx 2^{65000}$$

Perfect similar alignments

- similar alignments in $\mathbb{R}^3$ (Boxer 1999):

$$O(n^3 + \underbrace{kn^{2,2}}_{\text{output}} \log n)$$

- searching approximate alignments and/or partial alignments is a much more complex problem

Perfect alignments

ideas

- choose one triangle, e.g. (p_0, p_1, p_2), of P
- search for all congruent triangles in S
  (and their corresponding transformations)
- verify the rest of the points of P
  (after having applied the transformation)
- the run time is not proportional to n^3 (in case of congruence)
  because we can enumerate the triangles of S in a sophisticated manner and there are not as many possibilities

Approximate alignments

- as stated, we work in two steps
  - we search for adequate matchings μ (according to a certain tolerance)
  - we calculate the optimal transformations T (according to a certain distance measure)
- we select the best alignment(s)

Optimal Transformation

- let $S' = \mu(P)$ be a matching
- let d be a distance measure
- we look for the optimal rigid motion transformation T, (only translation and rotation), such that
- $d(T(P), \mu(P)) = d(P', S')$ is minimal

Root mean square distance

$$
\begin{aligned}
d &= \sqrt{\frac{1}{n}\sum_i d(p'_i, s'_i)^2} \\
&= \sqrt{\frac{1}{n}\sum_i (U \cdot p_i + t - s'_i)^2}
\end{aligned}
$$

- U 3x3 rotation matrix, i.e., orthonormal
- t translation vector

Objective: find U and t such that d is minimal

A little bit of math

we observe: t and U are independent

- with the partial derivative of d according t

$$\frac{\partial d}{\partial t} = 2 \cdot \sum_i (U \cdot p_i + t - s'_i) = 2U \sum_i p_i + 2nt - 2 \sum_i s'_i$$

we obtain

$$\begin{aligned} t &= -U \frac{1}{n} \sum_i p_i + \frac{1}{n} \sum_i s'_i \\ &= -U \cdot p_c + s'_c \end{aligned}$$

- where p_c and s'_c are the centroids of both sets

Optimal Rotation

- with the above, d can be written as

$$\begin{aligned} d &= \sqrt{\frac{1}{n}\sum_i (U \cdot p_i + t - s'_i)^2} \\ &= \sqrt{\frac{1}{n}\sum_i (U \cdot (p_i - p_c) - (s'_i - s'_c))^2} \end{aligned}$$

- where U is a matrix with restrictions (has to be orthonormal)

Extremal points of functions with restrictions

- one converts the problem with restrictions
- with the help of LAGRANGE multiplies into
- a problem without restrictions
- which exhibits the same extremal points

Let's skip the details

- basically, we calculate first and second derivative according to the entries u_{ij} of U
- we search for the extremal points
- KABSCH algorithm 1976, 1978
- open source code at my home page

KABSCH algorithm

- let $\mathbf{S}$ be the matrix of rows containing the s'_i
- let $\mathbf{P}$ be the matrix of rows containing the p_i
- we compute $\mathbf{R} = \mathbf{S} \cdot \mathbf{P}^\top$
- we set $\mathbf{A} = [\mathrm{a}_0\ \mathrm{a}_1\ \mathrm{a}_2]$ with a_k being the eigenvectors of $\mathbf{R}^\top \mathbf{R}$
- we compute $\mathbf{B} = [\|\mathbf{R}\mathrm{a}_0\|\ \|\mathbf{R}\mathrm{a}_1\|\ \|\mathbf{R}\mathrm{a}_2\|]$
- and finally, we get $U = \mathbf{B} \cdot \mathbf{A}^\top$

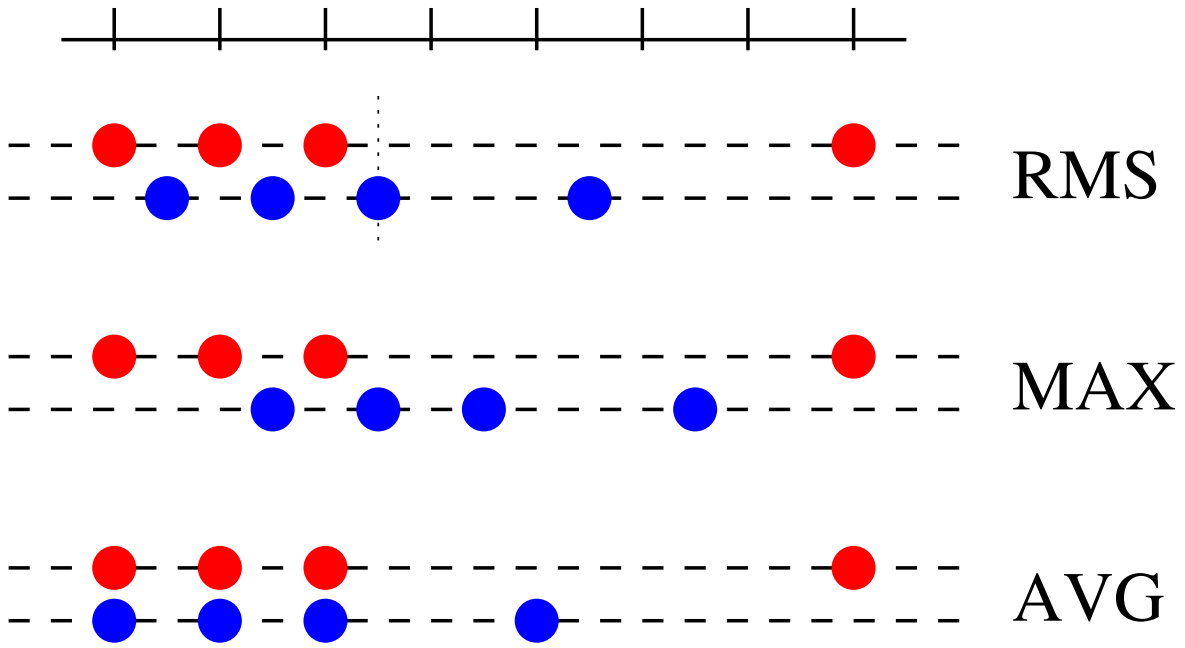
Introduction of scaling

- let us introduce a scaling value $\sigma \in \mathbb{R}$

$$d = \sqrt{\frac{1}{n}\sum_i (\sigma U \cdot (p_i - p_c) - (s'_i - s'_c))^2}$$

- let $p''_i = U \cdot (p_i - p_c)$ be the translated and rotated point p_i
- let $s''_i = s'_i - s'_c$ be the centralized point s'_i
- the solution for the optimal σ: $\sigma = \dfrac{\sum_i \langle s''_i, p''_i \rangle}{\sum_i \langle p''_i, p''_i \rangle}$

Different distance measures—different alignments



Optimal transformations for non–derivable distance measures

- if the function for d is not derivable, e.g., the average
- we use a gradient free optimization method
  (only with evaluations of the function)
- recently developed iterative method that is guaranteed to converge towards a local minimum
- algorithm of RODRÍGUEZ/GARCÍA–PALOMARES (2002)

Idea behind RODRIGUEZ/GARCIA–PALOMARES

- let $f(\mathbf{x})$ be the function to be minimized
- we iterate contracting and expanding adequately parameters $h^k > 0$ and $\tau > 0$ such that
- $f(\mathbf{x}_{i+1}) = f(\mathbf{x}_i \pm h^k \mathbf{d}_k) \leq f(\mathbf{x}_i) - \tau^2$
- where $\mathbf{d}_k$ is a direction taken from a finite set of directions (which depends on the point $\mathbf{x}_i$)
- with $\tau \longrightarrow 0$, $\mathbf{x}_i$ converges to local optimum (while there are no constraints)

Rotation in *quaternion space*

- a rotation $U \cdot p$ of the point p with the matrix U can be expressed as
- $q \star \bar{p} \star q^{-1}$ in quaternion space $\mathbb{H}$
  (HAMILTON formula, $\mathbb{C} \sim \mathbb{R}^2$, $\mathbb{H} \sim \mathbb{R}^4$)
- where $\bar{p} = (0, p)$ is the canonical quaternion of the point p
- and $q = (\sin(\varphi/2), \cos(\varphi/2)u)$ is the rotation quaternion
  (with $u \in \mathbb{R}^3$ being the axis and φ the angle of rotation)
- instead of U with 9 constraint variables
  we have u and φ, i.e., 4 unconstraint variables

Matching Algorithms

1. maximal clique detection within the graph of compatible distances
2. geometric hashing of the pattern
3. distance geometry

Maximal clique detection

- we generate a graph $G = (V, E)$ (graph of compatible distances)
- vertices $v_{ij} \in V$ all pairs (p_i, s_j)
- edges $e = (v_{ij}, v_{kl}) \in E$, if $d(p_i, p_k) \approx d(s_j, s_l)$
- search for maximum cliques in G

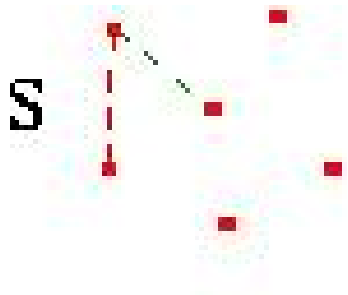
Properties of Clique Detection

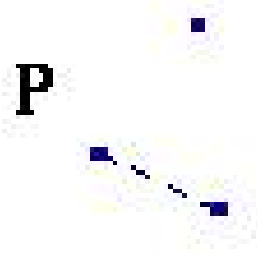
- the problem is NP–complete
  (however, we search only for *cliques* of size $\leq k$)
- fast algorithms need adjacency matrices
- if $n = |S| = 5000$ and $k = |P| = 100$ we need 30 GByte
  (counting only one bit per edge)

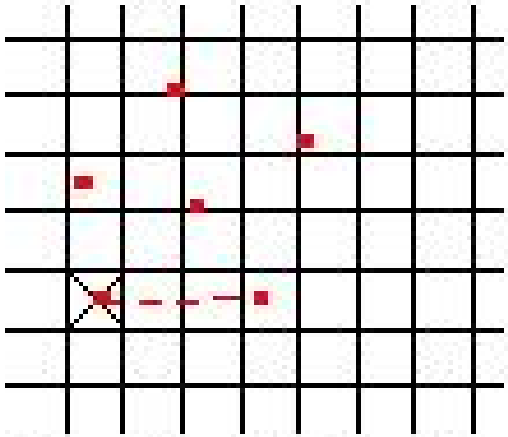
Geometric hashing

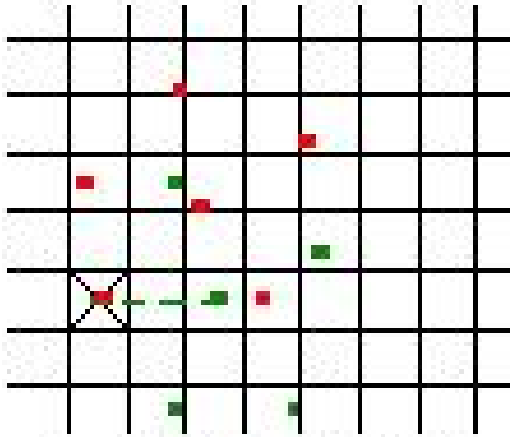
- preprocessing of the search space
- let's describe the two–dimensional case
  - we align each pair (s_i, s_j)
    with s_i at the origin and s_j in direction x
  - we insert some information for each other point $s_k \in S$
    in a hashtable defined on a grid over S

Example: geometric hashing









Searching with geometric hashing

- we simulate an insertion of the points of P into the hashtable
- but we count only the non–empty entries
- many votes reveal candidates for partial alignments
- e.g., if we encounter a pair (p_i, p_j) such that
  for each other point of the pattern there is a non–empty cell in the hashtable
  we have found a perfect candidate

Properties of geometric hashing

- grid size must be selected beforehand
- preprocessing time $O(n^{d+1})$
- searching time $O(k^{d+1})$
- works only for rigid motion transformations

Distance geometry

- we represent both sets S and P as distance graphs
- the vertices of the graphs are the points of the sets
- the edges of the graphs hold the distances between the corresponding vertices
- e.g., $G_P = (P, P \times P)$ complete graph

The ideas behind psm

- we define adequate distance graphs G_P and G_S
- we search for subgraphs G'_S of G_S that are congruent to the graph G_P
  (allowing certain tolerances)
- we optimally align G_P with the subgraphs of G'_S
- we select the best one among all hits

- we extend the search to work with subgraphs of G_P as well
- we select a best subgraph as final solution

The four main steps of psm

- construction of the graphs
  with: exploitation of locallity properties
- search of subgraphs
  with: sophisticated backtracking
- alignment
  with: minimization of cost functions
- search of partial patterns
  with: reactive tabu search

Construction of the graphs

- let us assume that the pattern P is small
- we construct G_P as the complete graph
- we generate a dictionary D
  (ordered data structure)
  that contains all distances (intervals) between points in P
- we consider an edge between two vertices in G_S
  if the distance is present in the dictionary D

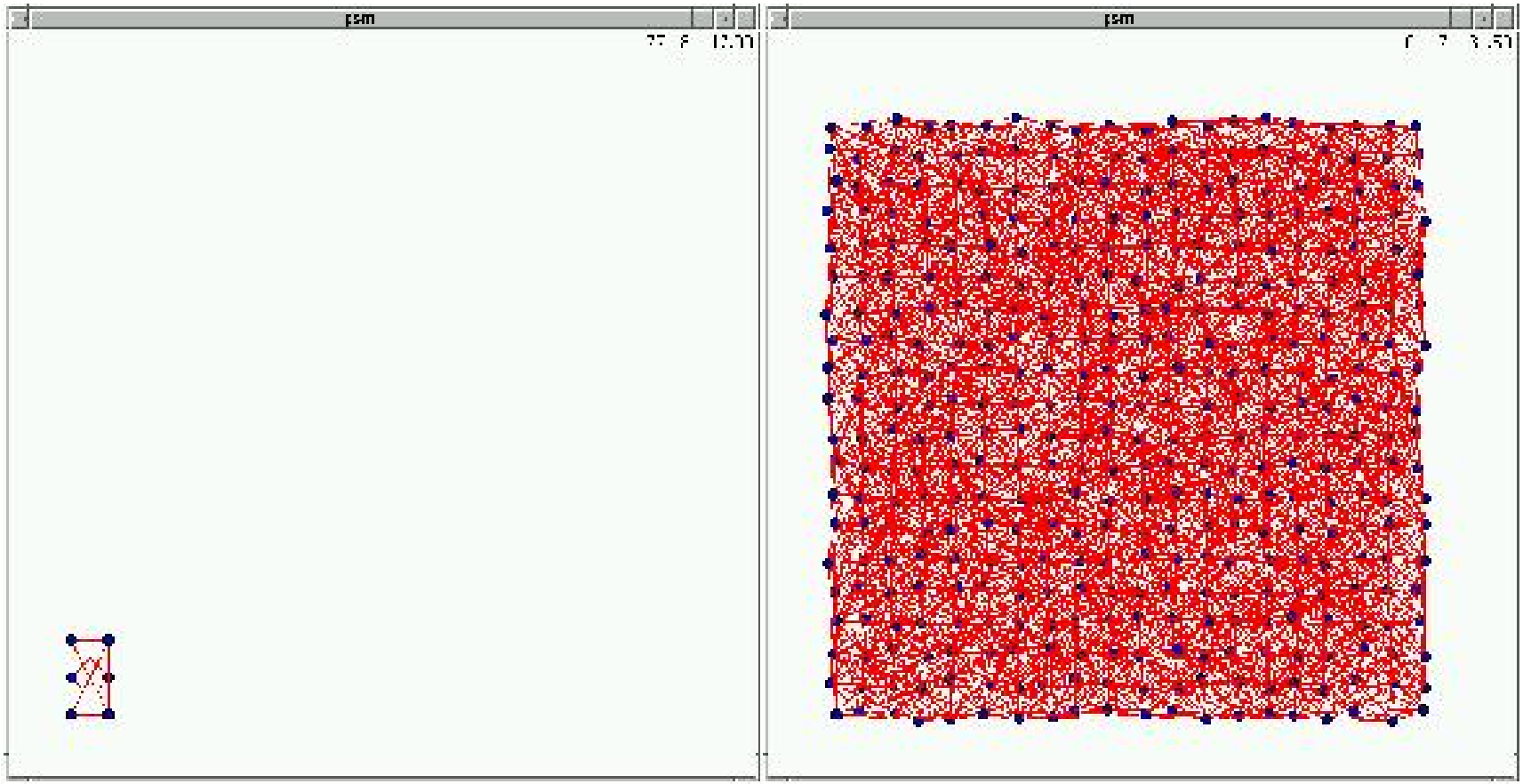
Construction of the dictionary

- let $d_{ij} = d(p_i, p_j)$ be the distance between two points of P
- the dictionary will contain the interval

$$[\quad (1-\varepsilon)\cdot d_{ij} \quad , \quad (1+\varepsilon)/(1-\varepsilon)\cdot d_{ij} \quad] \quad \in D$$

  where $0 \leq \varepsilon < 1$ is an appropriate tolerance
- the upper limit can be simplified to $(1+\varepsilon)$
  (but we loose the symmetry)
- we can join intervals in the dictionary D if they intersect

Construction of the graphs that way

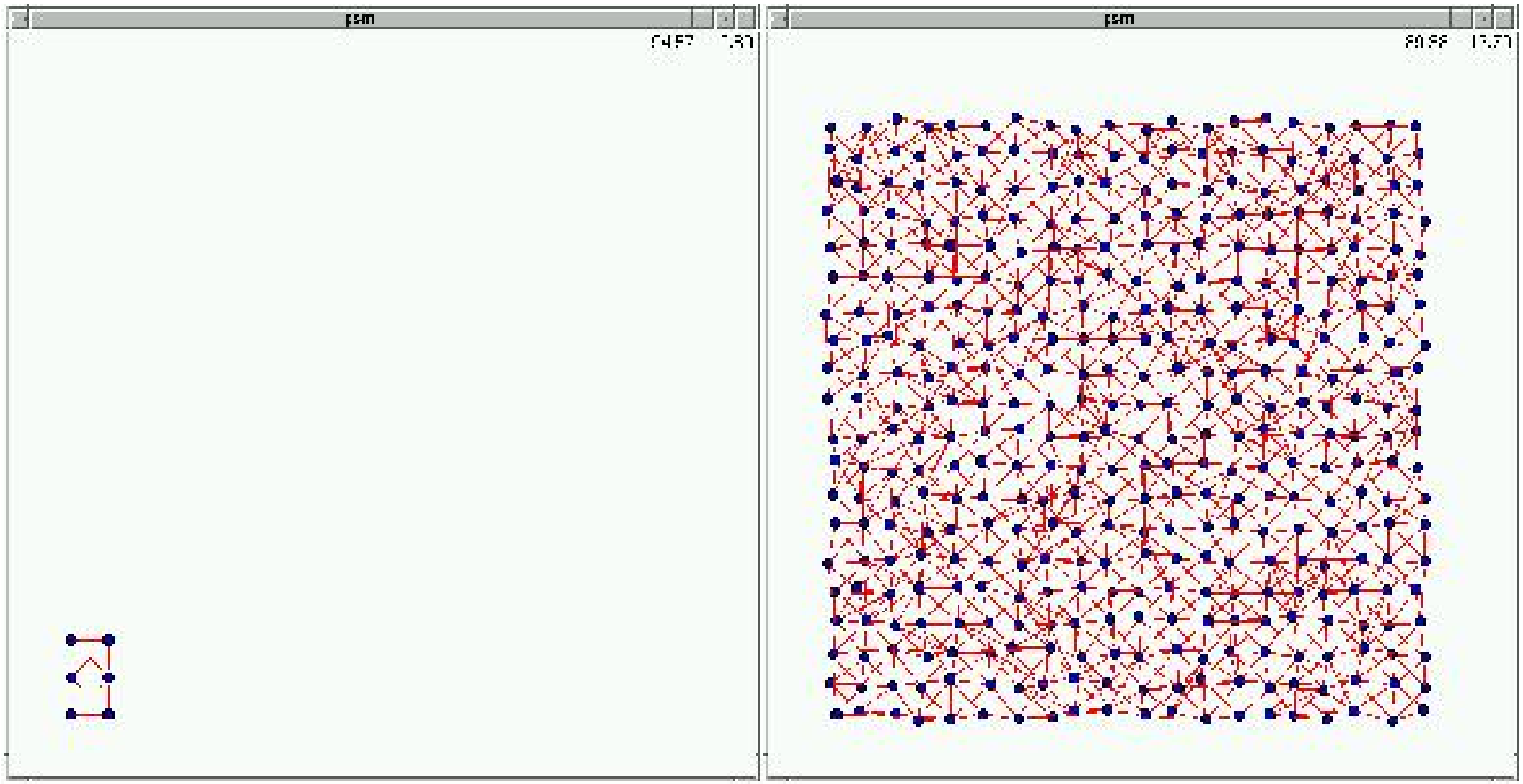


G_P G_S

Fast construction of the graphs

- we construct G_P as a connected (and rigid) graph mantaining only the short edges
- we order the points of S previously in a grid of size similar to the largest of the intervals

Construction of the graphs that way



G_P G_S

Matching

- let us assume (at the beginning) that G_P is a complete graph
- we order the points of G_P according to any order e.g. $(p_0, \ldots, p_{k-1})$
- we apply a backtracking algorithm that tries to encounter for earch p_i a partner s_i following the established ordering
- hence:

Backtracking for the search

- let us assume that we already found a subgraph $G_{s_0,\ldots,s_i}$ where the graph $G_{p_0,\ldots,p_i}$ can be matched
- we look for candidates s_{i+1} for the next point p_{i+1}
  - that must be neighbors of the point s_i within G_S
  - that must not be matched already and
  - that have similar distances to the s_j ($j \leq i$) as the p_{i+1} to the p_j ($j \leq i$)
- while there is a candidate we advance with i
- if there are no more candidates for s_{i+1}, s_i cannot be a partner for p_i neither (i.e.: backtracking)

Termination of the algorithm

- the algorithm *informs* each time
  a candidate for for p_{k-1} has been found
- the algorithm terminates when
  - there are no more candidates for p_0 or
  - the first solution has been found

Optimizations of the basic algorithm

- reduction of the edges in G_P
  implies: reduction of the edges in G_S
- *good* ordering of the p_i
  implies: reduction of the number of candidates
- consideration of the type of point (e.g. element type of the atom)
  implies: reduction of the number of candidates
- all heuristics imply:
  the backtracking advances faster

Search for partial alignments

- find the subset of the points of the pattern that can be matched best to some points in the search space
- NP–complete
- there are $|\mathscr{P}(P)| = 2^k$ possibilities to choose a subset
- we apply:
  - genetic algorithm
  - reactive tabu search

Genetic Algorithm

- maintain graph G_S as complete graph
- genome: sequence of bits indicating if a point belongs to the actual pattern or not
- crossover: *two point crossover*
- mutation: *flip*
- selection: roulette wheel
- cost function: distance and size of alignment

Termination of the GA

- it is not that easy
- once the first solution has been found
- once a sufficently good solution has been found
- after a certain number of iterations
- once diversity of population is too low

Problems with GA

- G_S must be a complete graph
  You know a crossover operation for non–complete graphs?
- more precisely:
  we need a crossover (and mutation) operation that maintains a specific property of the graphs (e.g., connectivity, rigidness)
- or some new idea...

Reactive Tabu Search

- we start with an admissible solution
- we search for possibilities to improve the current solution
- if we can: we choose one randomly
- if we cannot:
  - we search for possibilities to reduce the current solution
  - if we can: we again try improvements
  - if we cannot: we jump to another admissible solution

The Tabu criterion

- we avoid repetitive movements taking advantage of a memory that stores intermediate solutions
- i.e.: we mark certain movements as tabu for a certain number of iterations
- reactive means: we adapt the tabu period dynamically

Quality of a partial alignment

- evaluation of the cost of a solution:
  number of aligned points plus quality of the alignment
- remember: quality $Q \in]0, 1]$, but we will use $Q \geq$ threshold
- hence, maximal quality: $|P| + 1$

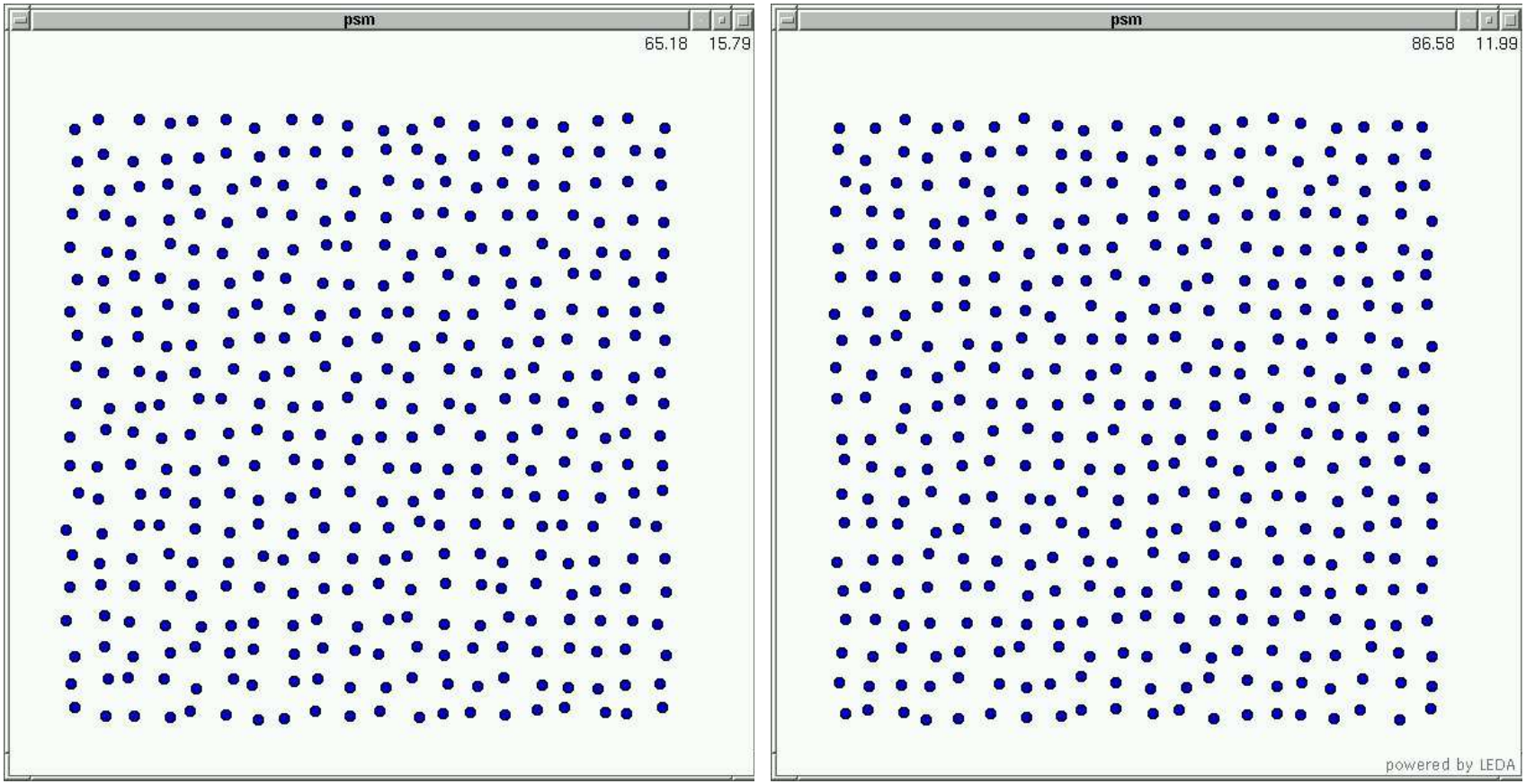
Reactive Tabu Search for `psm`

- representation of the problem:
  sets of indices of the matched points
- search for candidates to improve (*add*):
  (rigidly) connected neighbors within graph G_S
- search for candidates to reduce (*drop*):
  any point of the current solution that mantains the graph G_S connected (and rigid)

When do we terminate?

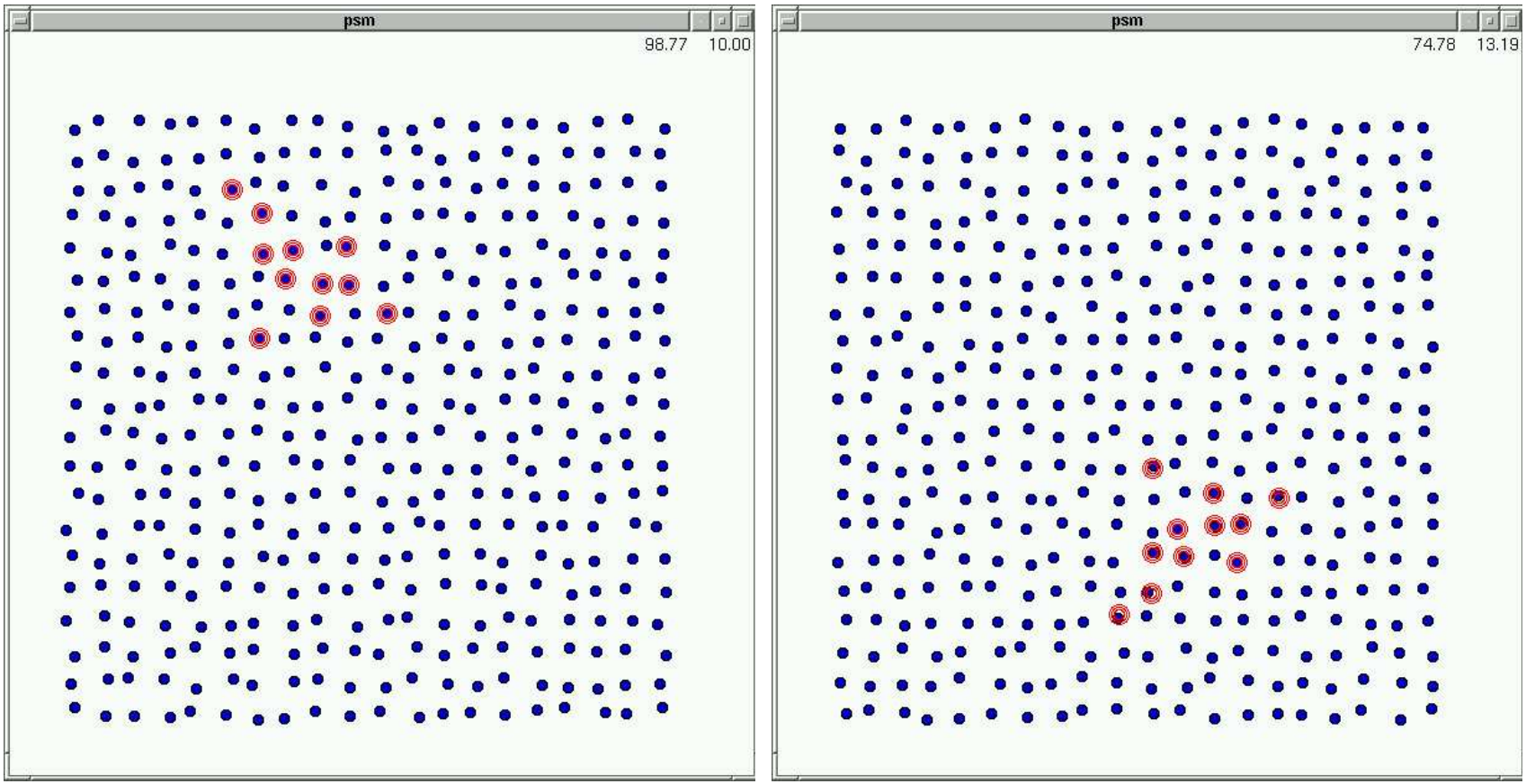
- not that simple
- once we found a sufficiently good solution
- once we have run a certain number of iterations

Search for the largest common pattern



P

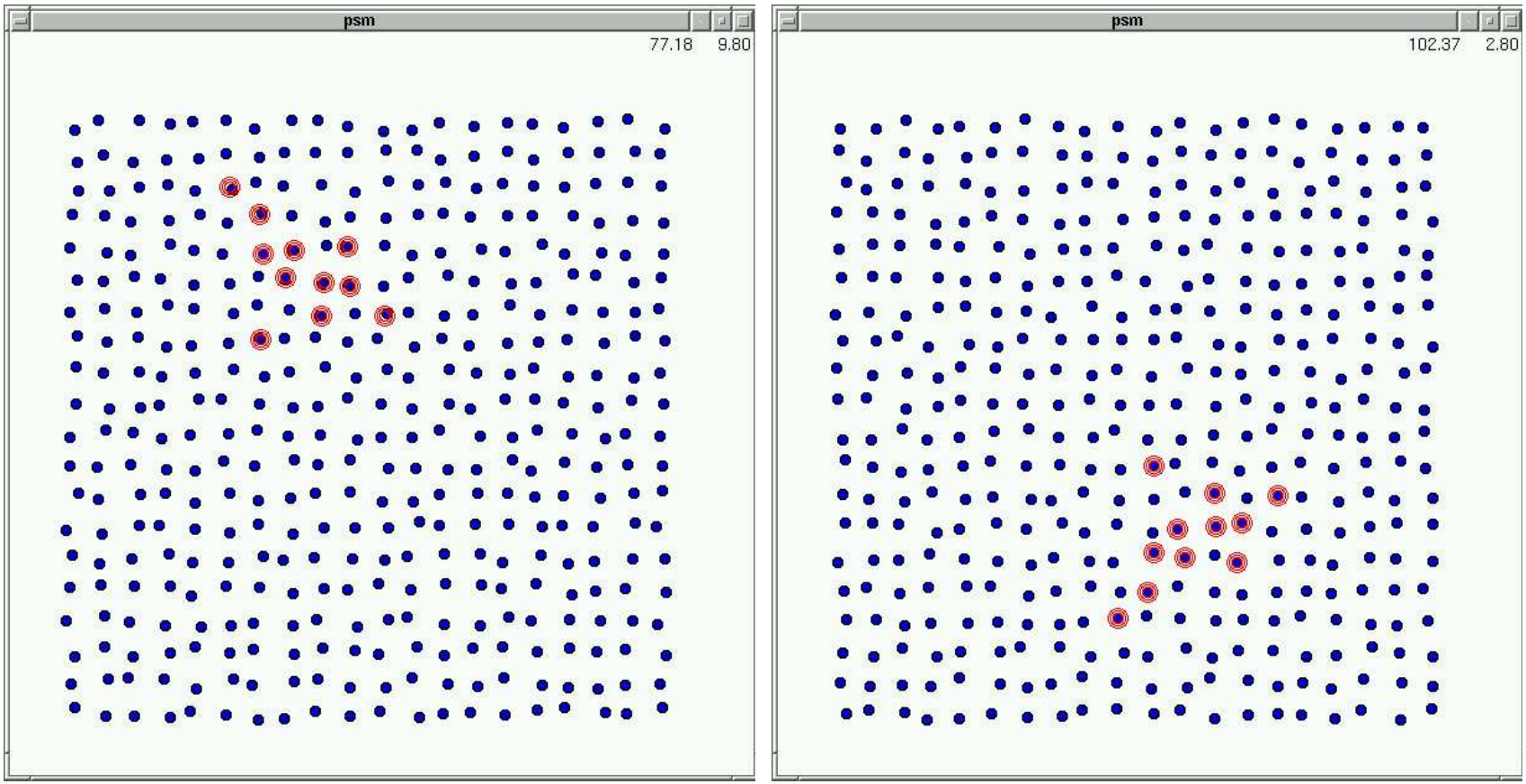
Search for the largest common pattern



p/P

p/S^2

Search for the largest common pattern



Search for pattern with deformation

- instead of the complete graph use a connected sparse graph
- parts of the graph could be rigid
- the graph may specify hinges or torsion axis

- command line tool with configuration file
- GUI
- web–site to perform searches

Possible extensions

- enumerate more rigorously all locations
  (up to now we have concentrated on the best solution)
- extend the properties of the graphs defining deformations of the pattern (e.g. torsion of parts, restriction of angles)
- allow local tolerances (e.g. per edge), especially with preknowledge of the biochemical properties
- improve heuristics with statistical analysis of distributions of distances (*look for the unusual first*)
- improve the user interface
- more applications

No free lunch theorem

theory

Basically states:

The performance of all optimization algorithms amortized over all objective functions is always equal (in discrete spaces).

With consequence: no algorithm can outperform (in general) exhaustive search (or even random search).

No free lunch theorem

reality

Fortunely, we are not interested in optimizing *some* function, rather we like to optimize a *specific* one, i.e., an optimization algorithm is only useful in *his field* (because necessarily there are fields where its performance is very poor).

optimUMTS

Optimization of wireless UMTS networks

Joint work with

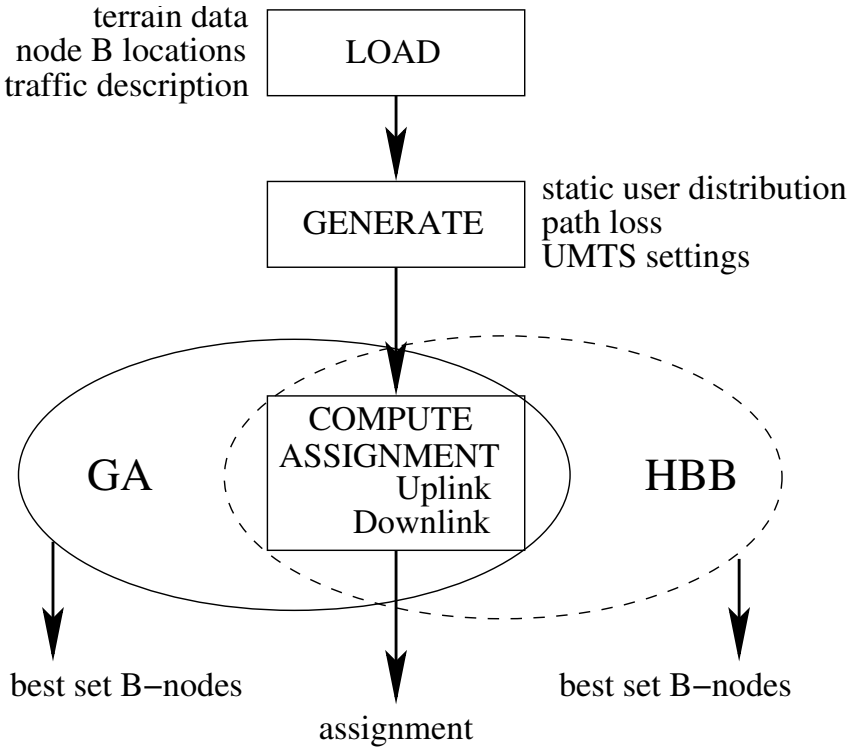
- Fernando Aguado
  Departamento de Teoría de la Señal
  Universidad de Vigo
- Luis Mendo
  Universidad Politécnica de Madrid

Objectives

- given a set of possible nodes B (base stations)
- find optimal subset
- to guarantee certain services (bandwidth)
- to an estimated user distribution

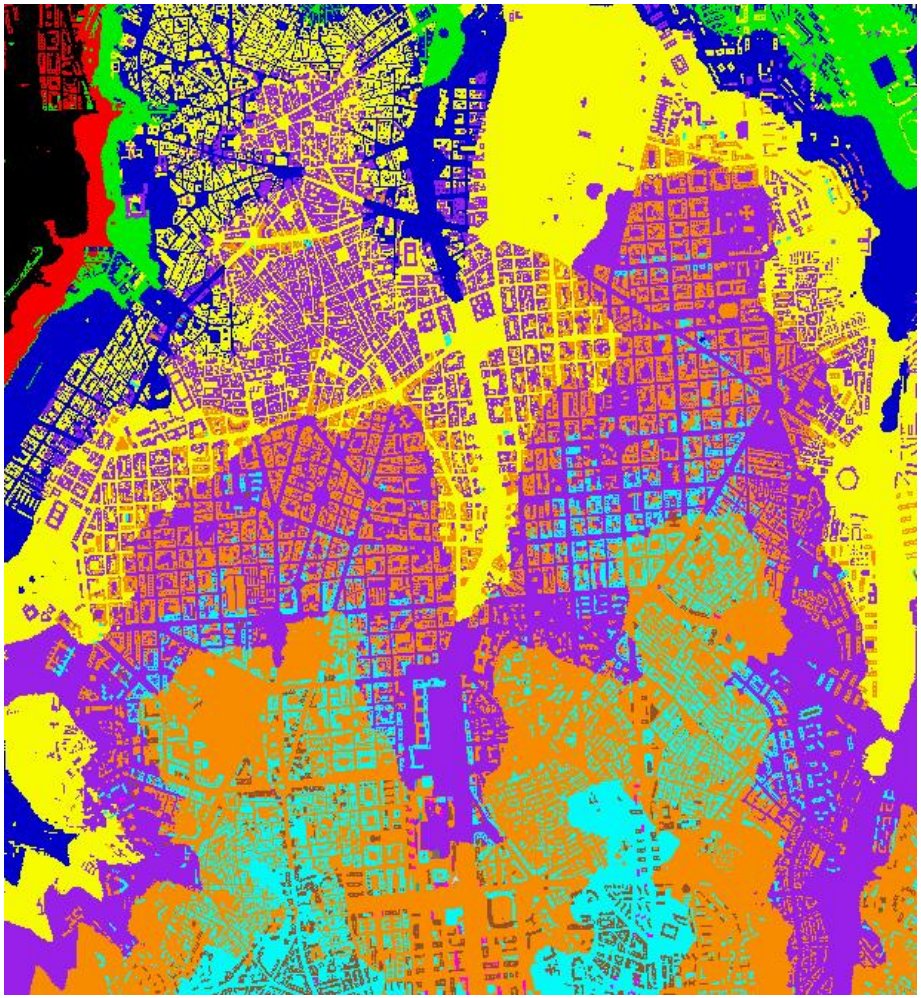
Principal algorithm

evolutionary and exact



Cartography of Madrid

input data



Calculation of attenuation matrix α

indirect input data

- simple logarithmic decay

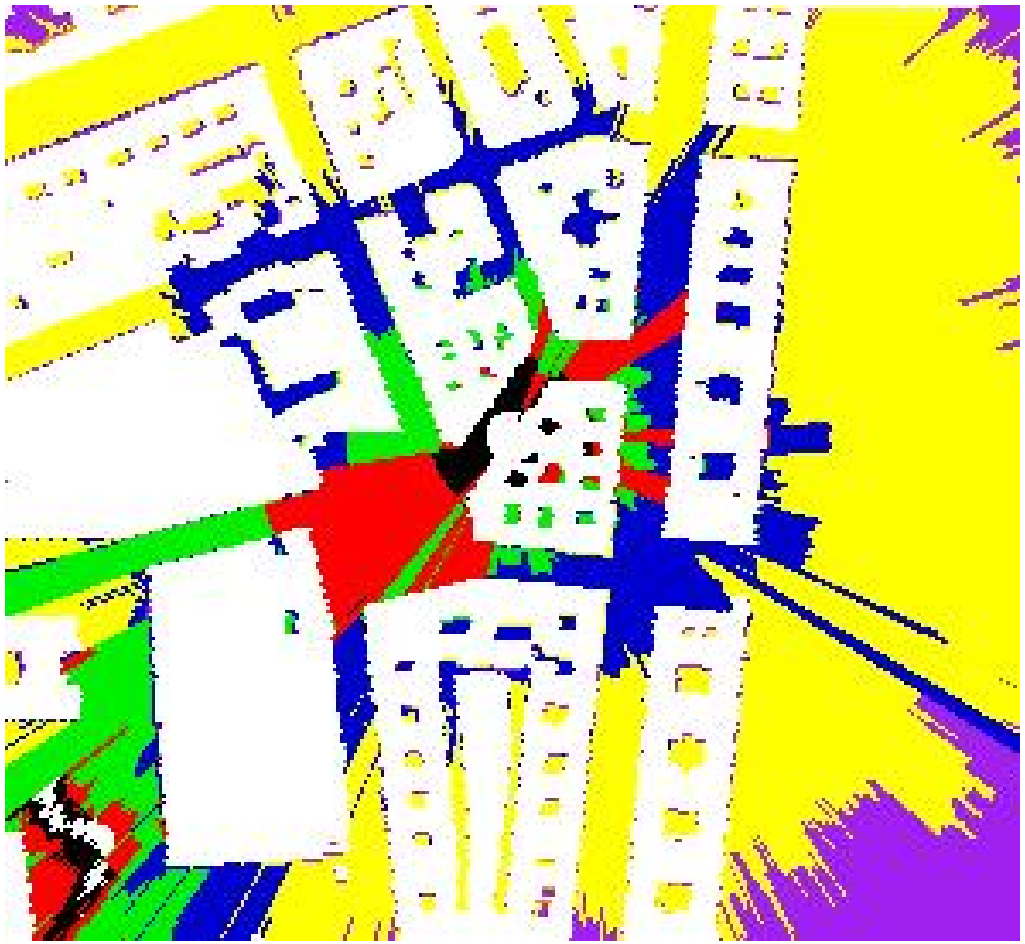
$$L(m,k) = 10^{0,1\cdot(32,2+35,1\cdot\log(d(m,k)))}$$

$$\alpha(m,k) = \frac{G_{\text{eff}}(m)\cdot G_{\text{eff}}(k)}{L(m,k)}$$

- simplied Xia model
- mixed model: close—Xia, far—simple

Coverage around a node B

computed input data



Traffic distribution

stochastic input data

- static distribution
- grid of estimated user with activation percentage
- polygons with Poisson process per service
- uniform distibution everywhere or only on streets

Calculation of SIR

objective function, part I

- uplink SIR γ_{UL}:

$$\gamma_{UL}(m,k) = (E_b/N_0)_{UL} \cdot b_0(k)/B_0$$

- downlink SIR γ_{DL}:

$$\gamma_{DL}(m,k) = (E_b/N_0)_{DL} \cdot b_0(k)/B_0$$

Calculation of noise

objective function, part II

- uplink noise $N_T(m)$ at transmitter m:

$$\begin{aligned} F(m) &= 10^{0,1 \cdot N^F(m)} \\ N(m) &= k_B \cdot T_{\text{amb}} \cdot B_0 \cdot F(m) \end{aligned}$$

- downlink noise $N_R(k)$ at receiver k:

$$\begin{aligned} F(k) &= 10^{0,1 \cdot N^F(k)} \\ N(k) &= k_B \cdot T_{\text{amb}} \cdot B_0 \cdot F(k) \end{aligned}$$

Calculation of power (Hanly and Mendo)

objective function, part IVa, iterative method

- initial power $P_0(k)$ for receiver k:

$$t_0(m,k) = \frac{\gamma_{UL}(m,k)}{\alpha(m,k)} \cdot N(m) \qquad P_0(k) = \min_m \{t(m,k)\}$$

- power $P_i(k)$ for receiver k in iteration i:

$$\begin{aligned} s_i &= P_{i-1} \cdot \alpha(m) \\ t_i(m,k) &= \frac{\gamma_{UL}(m,k)}{\alpha(m,k)} \cdot ((s_i - P_{i-1}(k) \cdot \alpha(m,k)) \cdot a(k) + N(m)) \\ P_i(k) &= \min_m \{t_i(m,k)\} \end{aligned}$$

Termination criteria for iteration

objective function, part IVb, iterative method

$$\exists k \text{ with } P(k) > P_{\max}(k) \implies \text{no assignment}$$

$$i > I_{\max} \implies \text{no assignment}$$

$$\max_k \left\{ \frac{P_i(k)}{P_{i-1}(k)}, \frac{P_{i-1}(k)}{P_i(k)} \right\} \leq \Delta \implies \text{assignment possible}$$

Calculation of assignment

objective function, part V

$$
\begin{aligned}
A(k) &= m \text{ with } t(m,k) = \min_{m}\{t(m,k)\} \\
P(k) &= \min_{m}\{t(m,k)\}
\end{aligned}
$$

Validation of assignment

objective function, part VI

- uplink:

$$P_{\min}(k) \quad \leq \quad P(k) \leq P_{\max}(k)$$

Direct solution

objective function, part VII

- downlink:

$$\beta(n,k,m) = \begin{cases} \rho(m,k) & \text{if} \quad m = n \\ 1 & \text{otherwise} \end{cases}$$

calculation of SSIR $\tilde{\gamma}$:

$$\tilde{\gamma}(m,k) = \frac{\gamma_{DL}(m,k)}{1+\rho(m,k)\cdot\gamma_{DL}(m,k)}$$

System to solve for transmitter power calculation

objective function, part VIII

$$
\begin{aligned}
H(m,n) &= \delta_{m,n} - \sum_{k \in \mathscr{A}^{-1}(m)} \frac{\alpha(n,k) \cdot \beta(n,k,m) \cdot \tilde{\gamma}(m,k)}{\alpha(m,k)} \\
\nu(m) &= P_{\text{plt}}(m) + \sum_{k \in \mathscr{A}^{-1}(m)} \frac{\tilde{\gamma}(m,k) \cdot N(k)}{\alpha(m,k)} \\
H \cdot T &= \nu
\end{aligned}
$$

Validation of power restrictions

restrictions I

- validation of maximum power of transmitter m:

$$0 \quad < \quad T(m) \leq T_{\max}(m)$$

Validation of power restrictions

restrictions II

- calculation of downlink power of receiver k:

$$P(k) = \frac{\tilde{\gamma}(m,k)}{\alpha(A(k),k)} \left(\sum_{m=1}^{M} \alpha(m,k) \cdot \beta(m,k,A(k)) \cdot T(m) + N(k) \right)$$

- validation of transmitter maximum channel power:

$$P(k) \leq P_{\text{chn}}$$

Optimization with heuristic backtracking

exact method

Observation:

- if there is no assignment with certain *m* nodes B
- then there is no assignment with less nodes B

hence

- start with all nodes B
- eliminate nodes B til optimum found with heuristic backtracking

Termination with heuristic backtracking

- stop if stagnation occurs
  (if within a subtree all minimum solutions are at the same depth)
- finds optimum solution

Heuristics

how to be fast at the beginning

- ordering of nodes B plays an important role in finding fast good solutions
- reorder nodes B for backtracking according to heuristics
- for instance: eccentricity, random, number of initial connections, etc.

Optimization with genetic algorithm (GA)

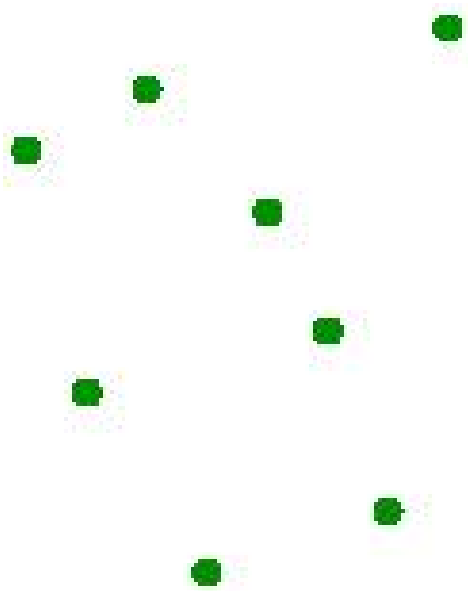
evolutionary method

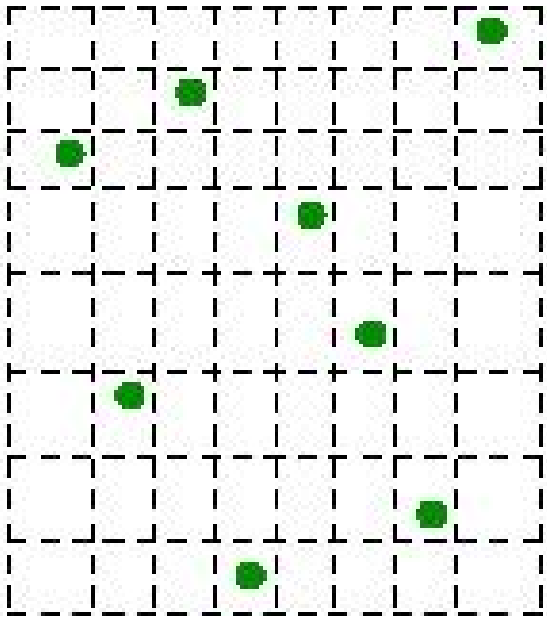
- steady-state incremental evolution
  (in each iteration two new decendents are generated)
- selection: roulette wheel
- mutation: flip
- crossover: two–point–cyclic
- quality: number of nodes B plus power as tiebreak

Genom generation I

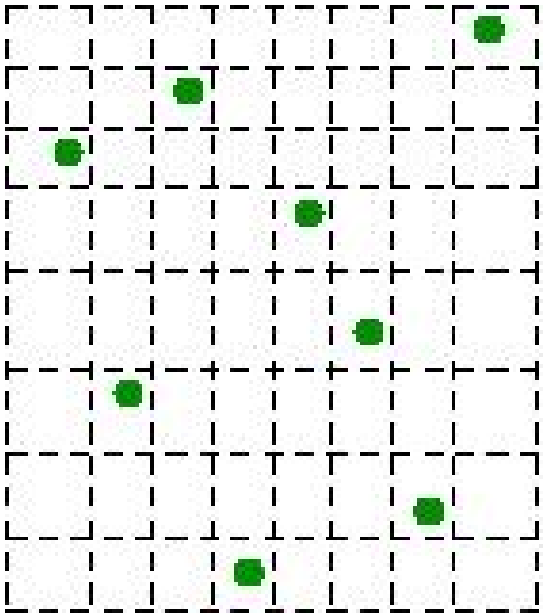
- matrix representation of nodes B
- exploiting locality properties
- using allele: usable, unusable, used, fixed

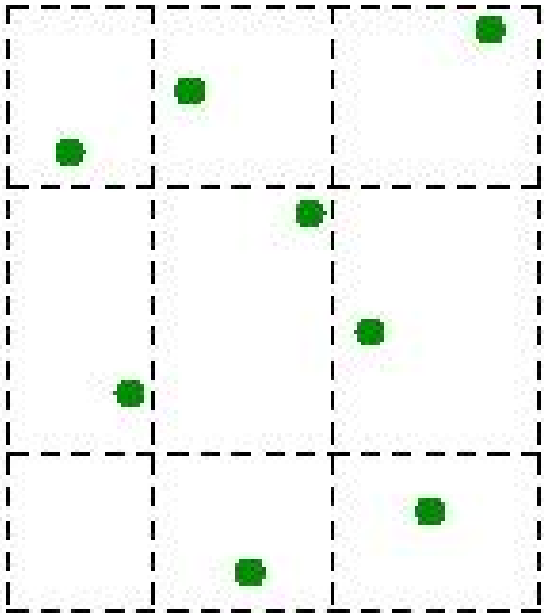
Genom generation II



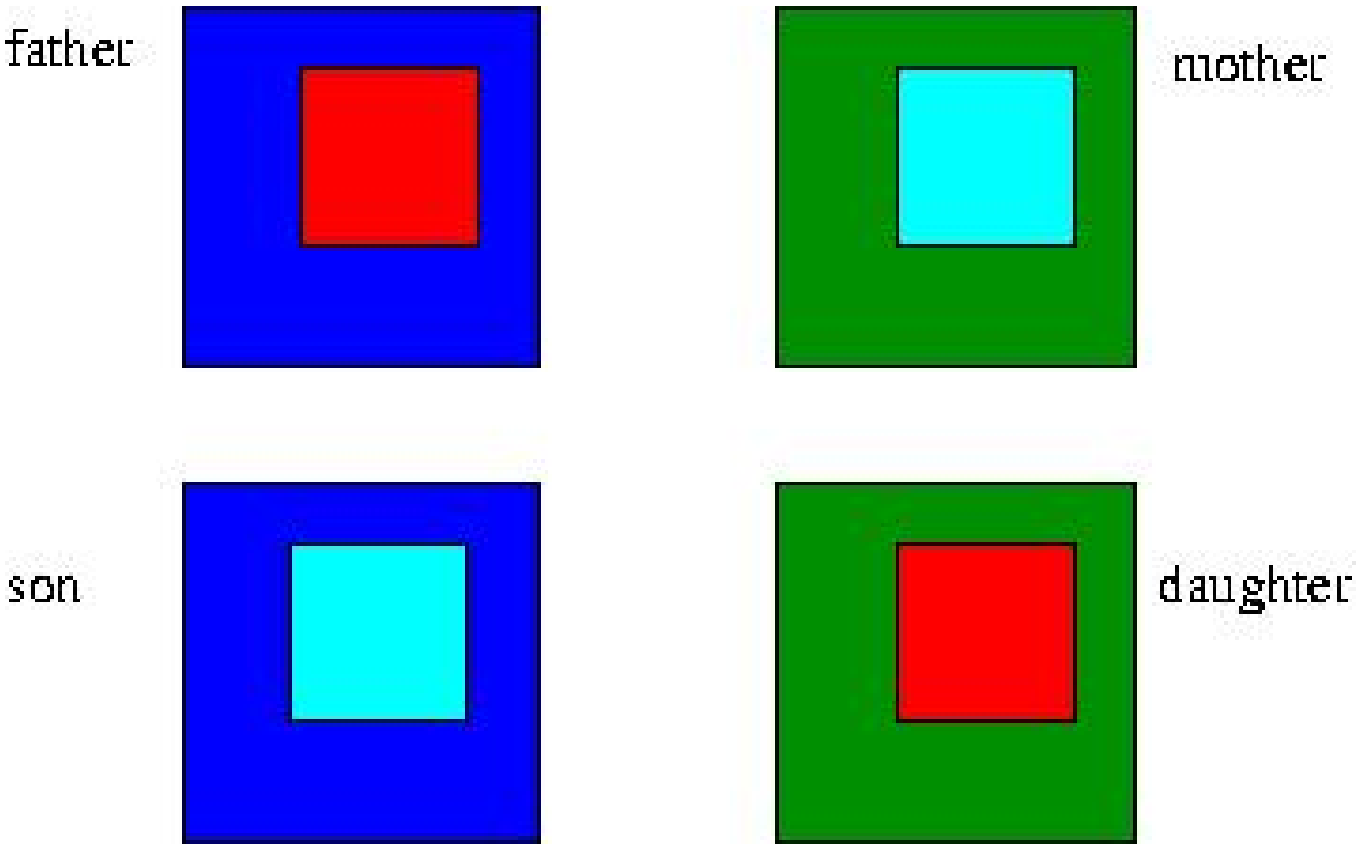


Genom generation III



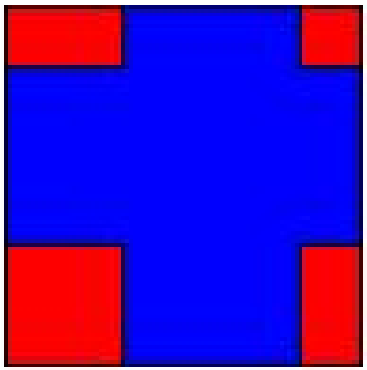


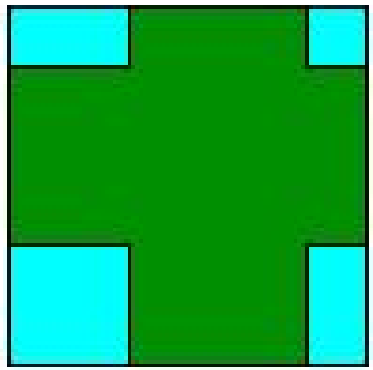
Crossover I



Crossover II

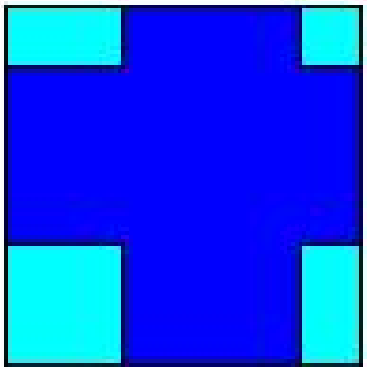
father

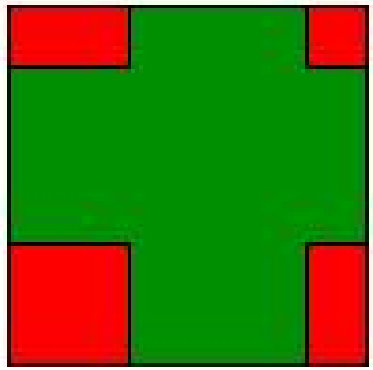




mother

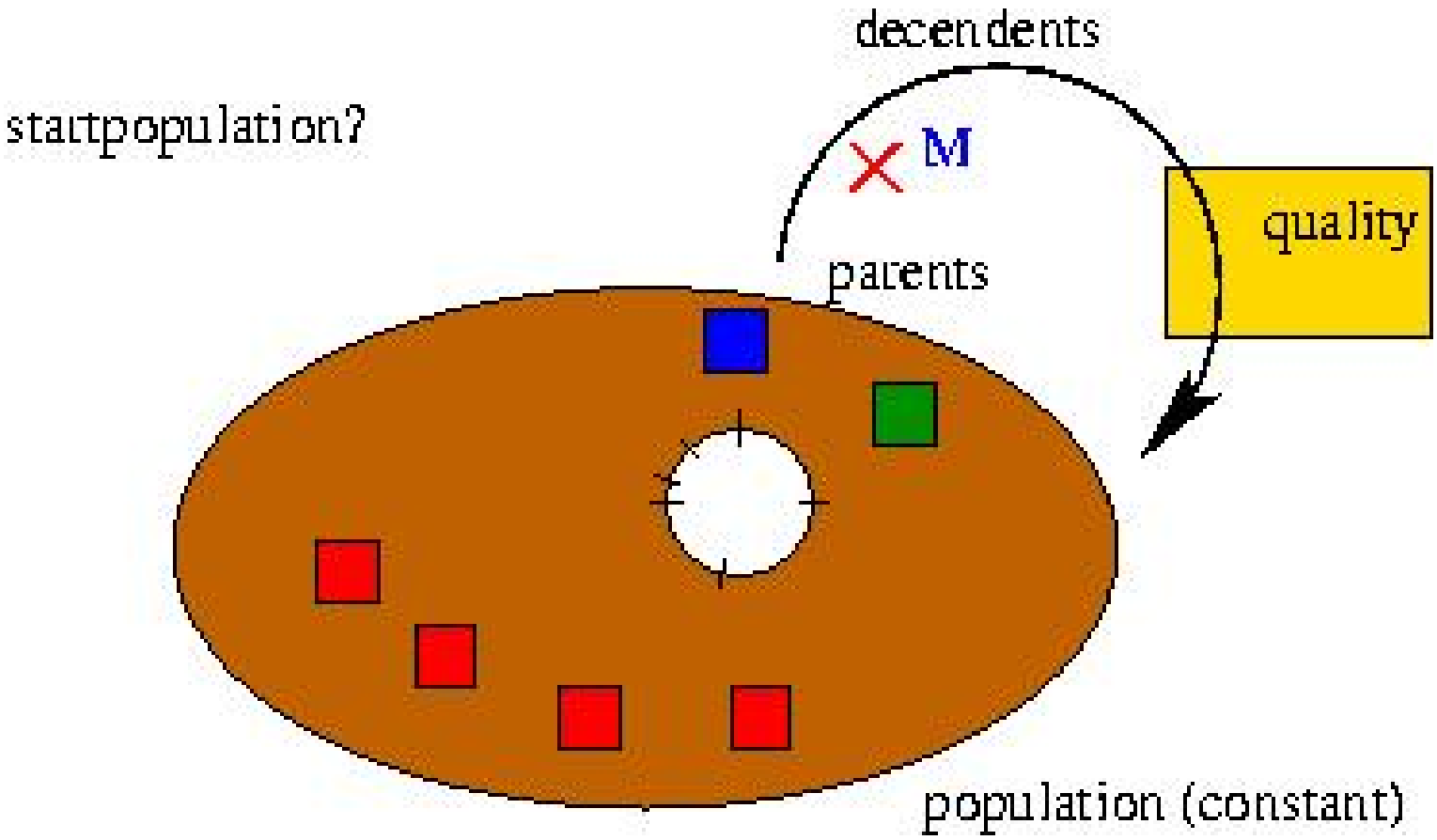
son



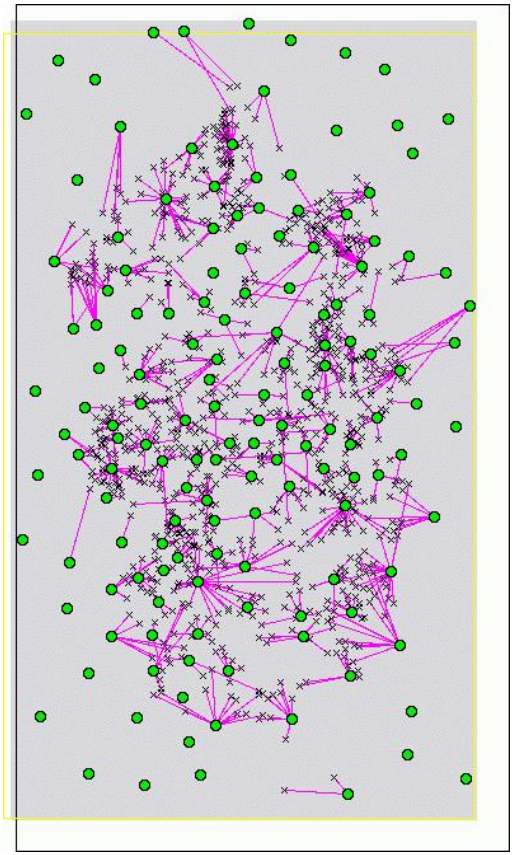


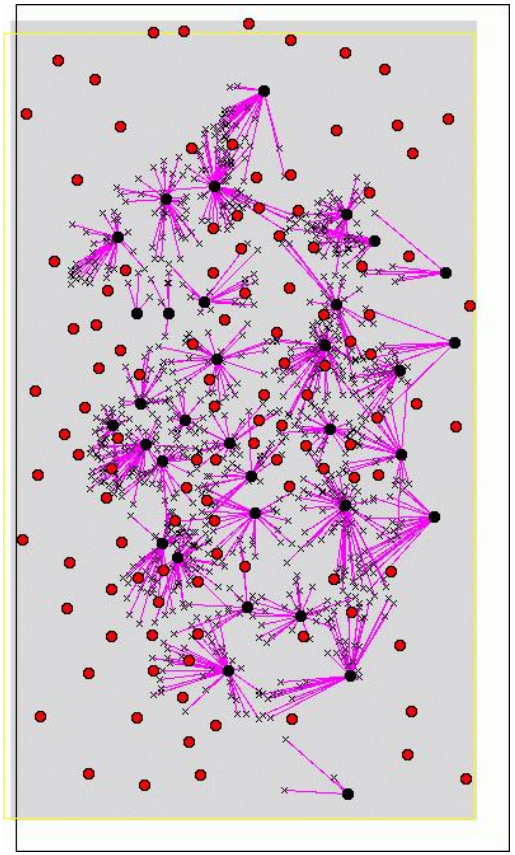
daughter

Evolution

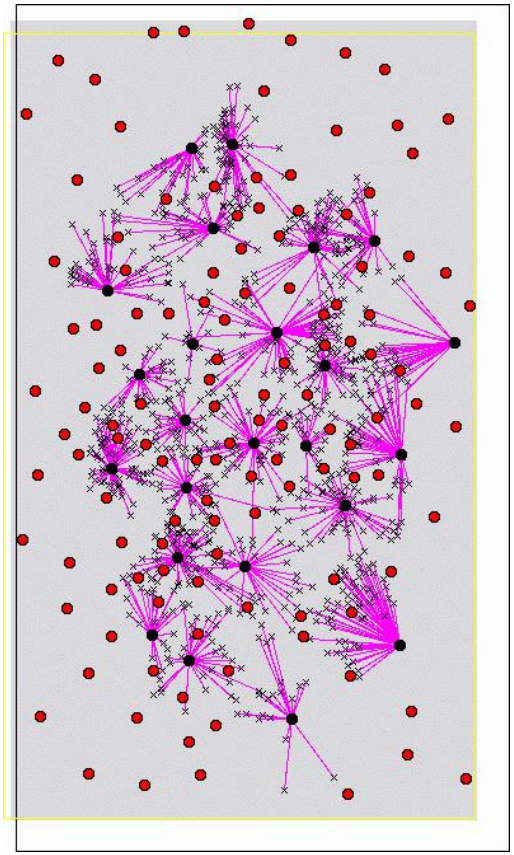

decendents
startpopulation?
M
quality
parents
population (constant)

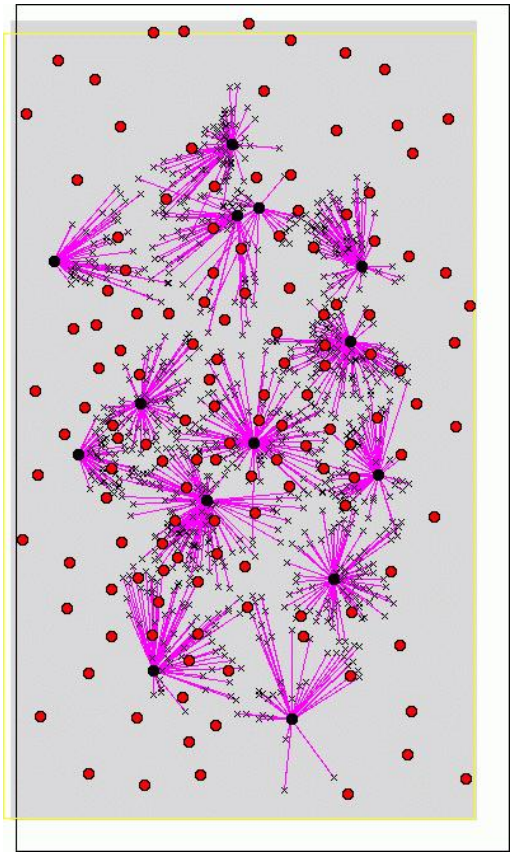
Assignment and heuristic branch result





GA results with and without downlink





Further research and implementations

- use of MonteCarlo method
- to find best subset of nodes B
- for several user distributions
- i.e., find best subset to satisfy different scenarios

Approximation of point sets with shapes

Informal problem description

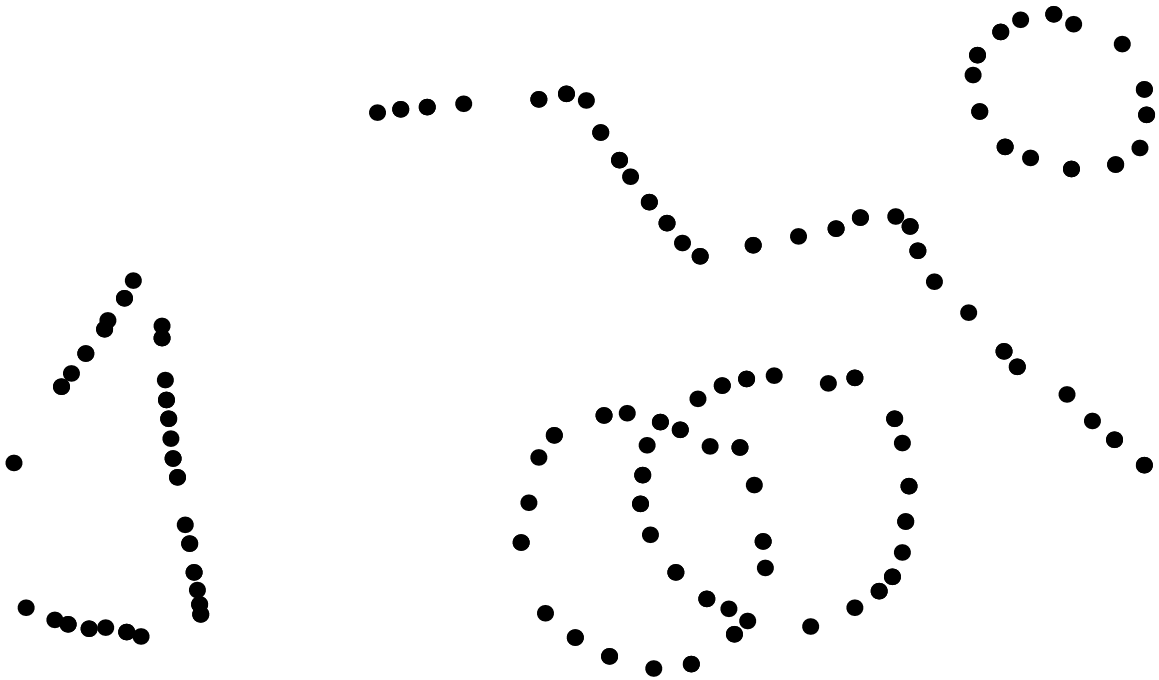
- given a set of points in the plane
- construct a geometric figure interpolating the sample points
- that reasonably captures the shape of the point set

Applications

- pattern recognition
- object definition in geographic information systems
- CAD/CAM services
- vectorization tasks
- curve reconstruction in image analysis
- single–computation pose estimation
- geometric indexing into pictorial databases
- shape tracking etc.

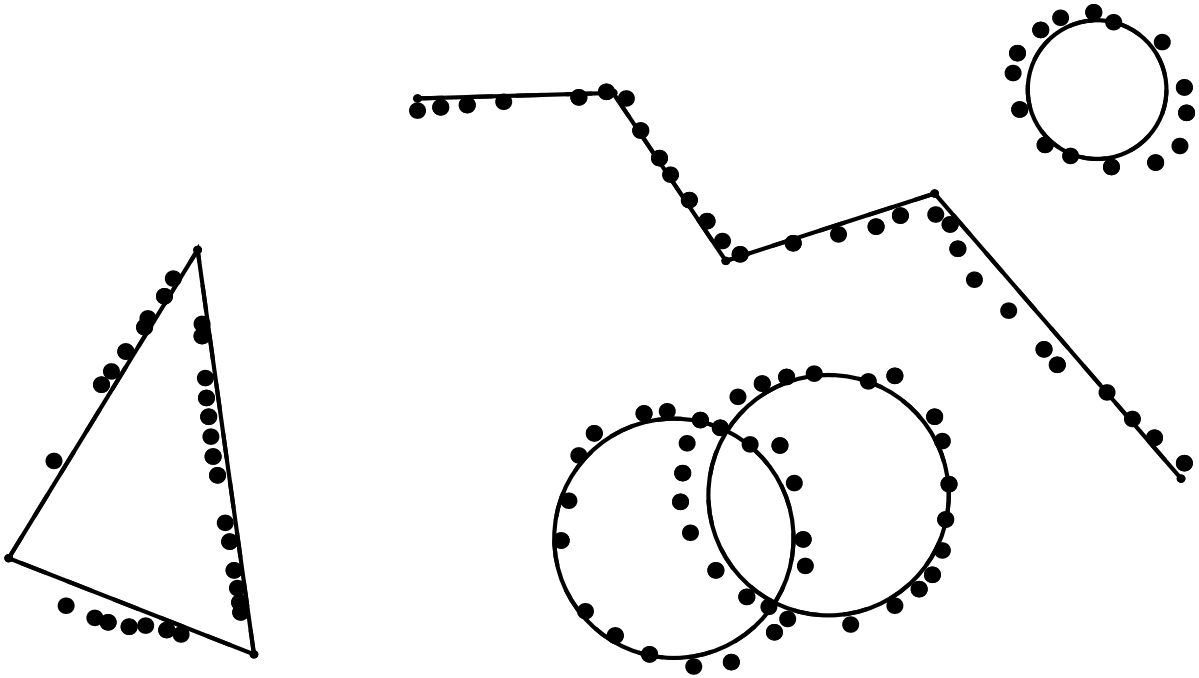
Example

point set



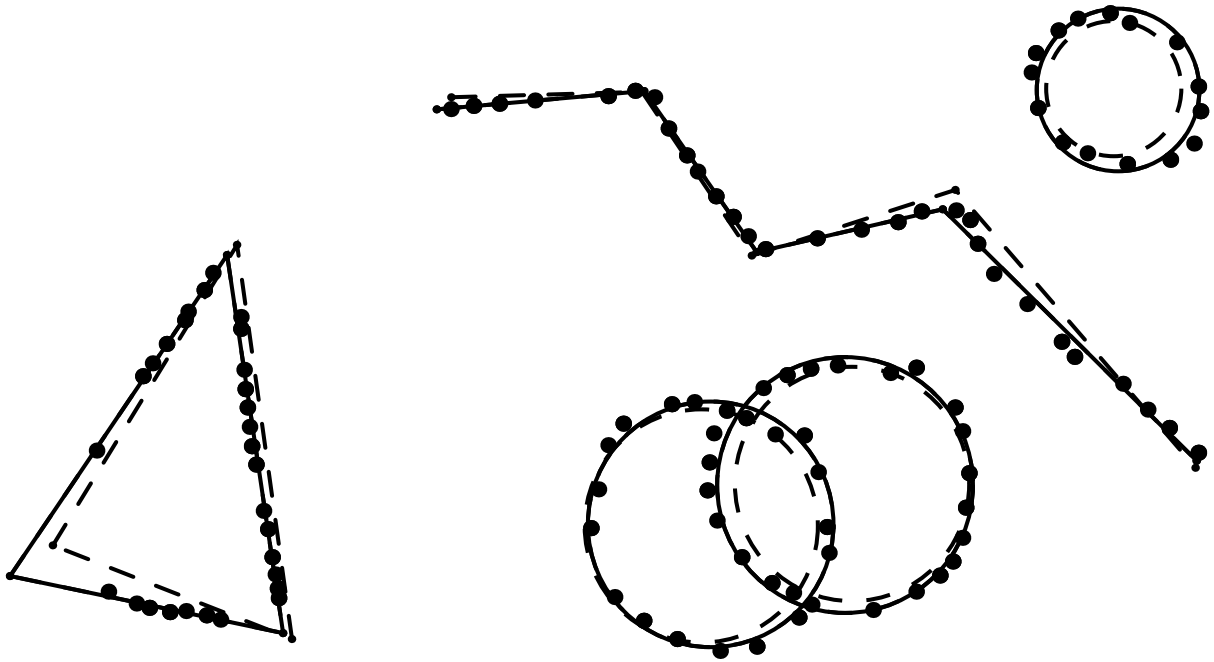
Example

initial shapes



Example

adapted shapes



Approximation task

Three steps:

- clustering of the points to identify the individual parts of a set of shapes,
- generating of an initial guess of the individual shapes,
- adapting the individual shapes to the underlying point set according to some distance metric.

Brain storming

distances, metrics, optimization, local and global minima, discrete–continuous, partially plain functions, multi–objetive optimization, local decisions, multi–scale, simplification, VORONOI–diagram, DELAUNAY–diagram, graph analysis, similarity detection, classification (with and without supervision), filtering

Polygonal approximation

type of clustering

- Given a set of points of a plane curve,
- construct a polygonal structure
- interpolating the sample points
- that reasonably captures the shape of the point set.

Three approaches for polygonal approximation

- α–shapes
- crust
- curve approximation

Alpha–shapes

application

The algorithm is able to detect

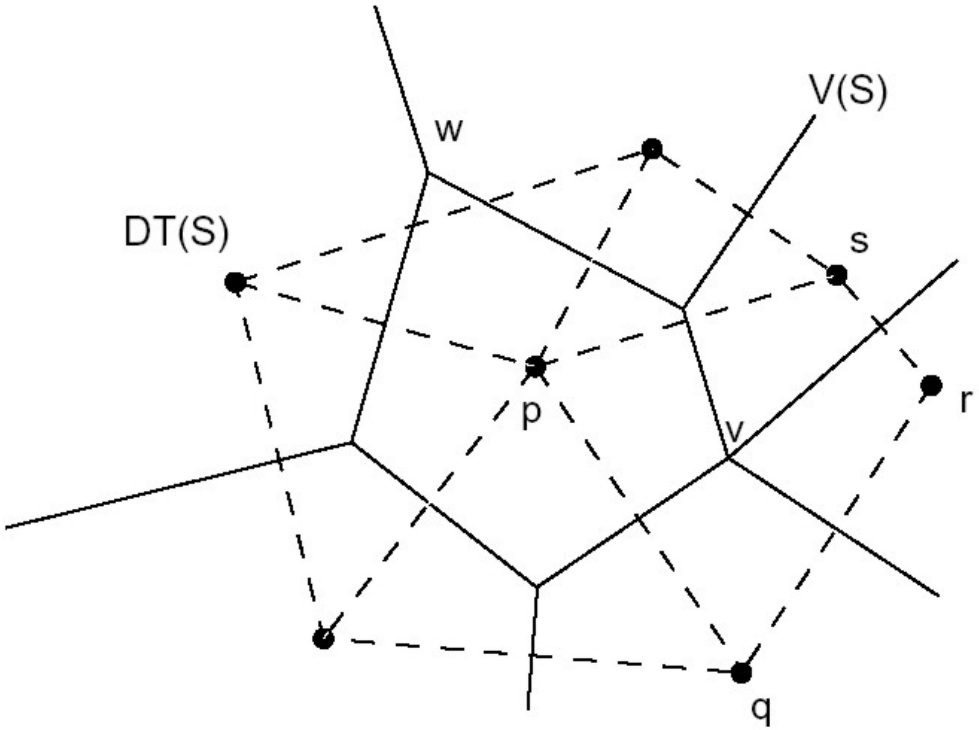
- the outer boundary of a set of points
- which covers more or less evenly distributed the interior of a shape.

Alpha–shapes

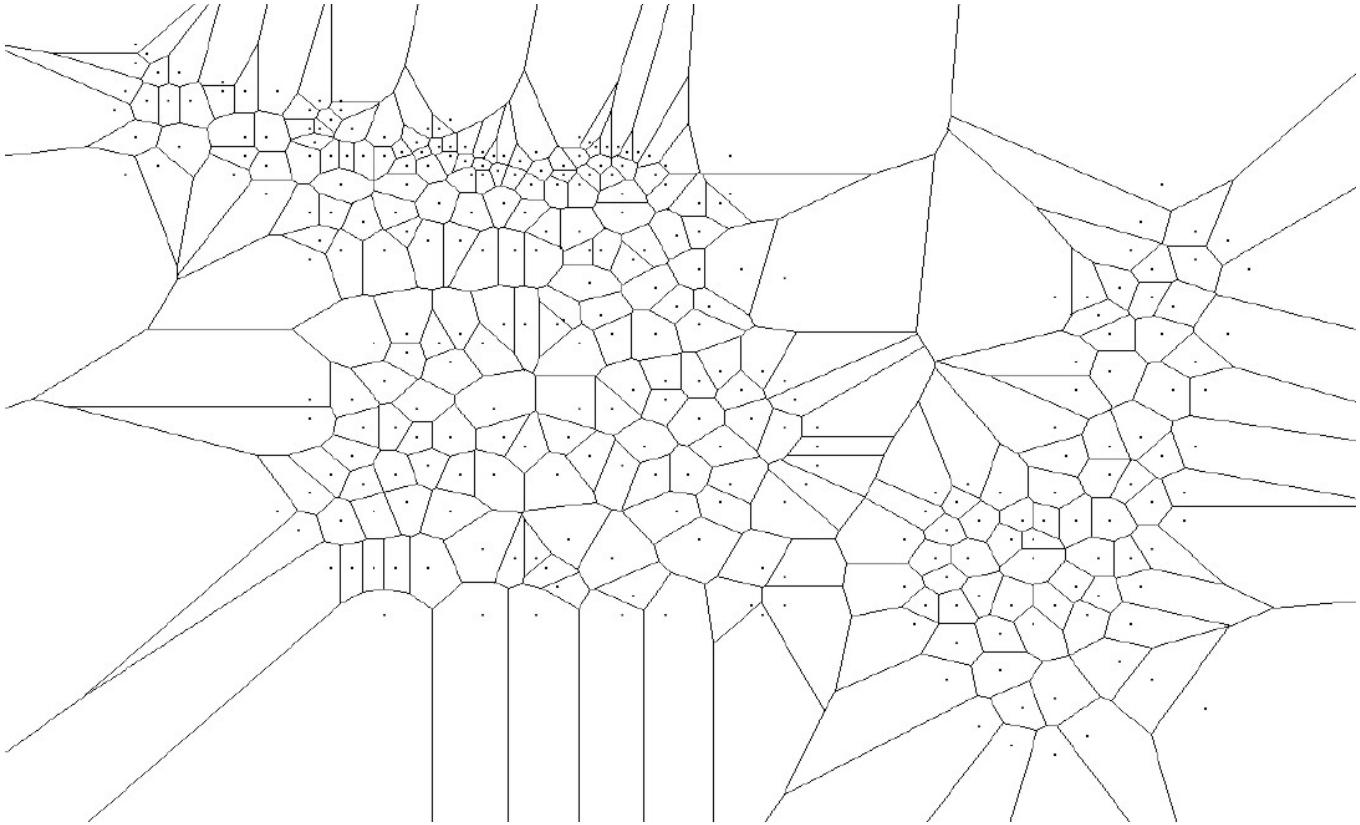
algorithm

- Compute the DELAUNAY triangulation of the point set.
- Eliminate all triangles of the resulting graph which have a radius larger than α times the minimum radius.
- The final shape is given by the outer edges of the remaining graph.

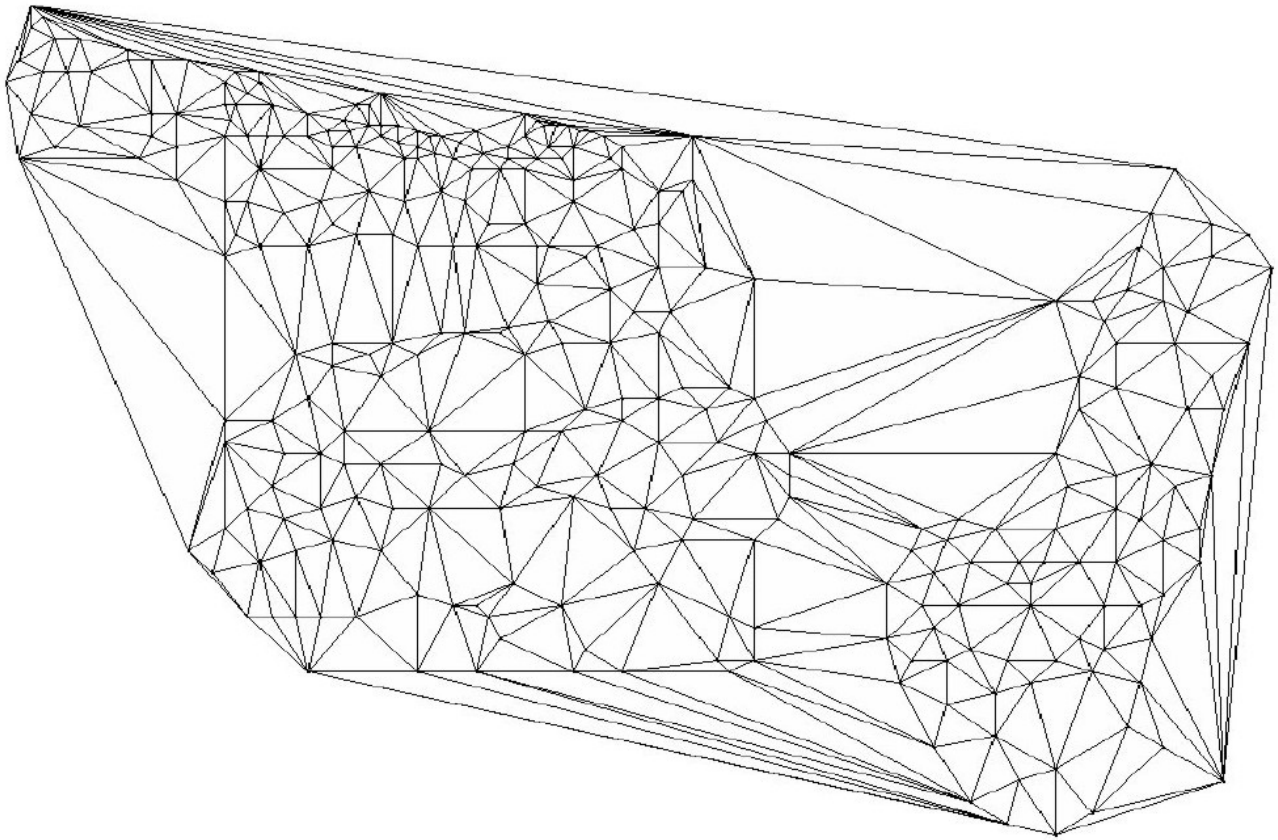
VORONOI–diagram



VORONOI–diagram with more points

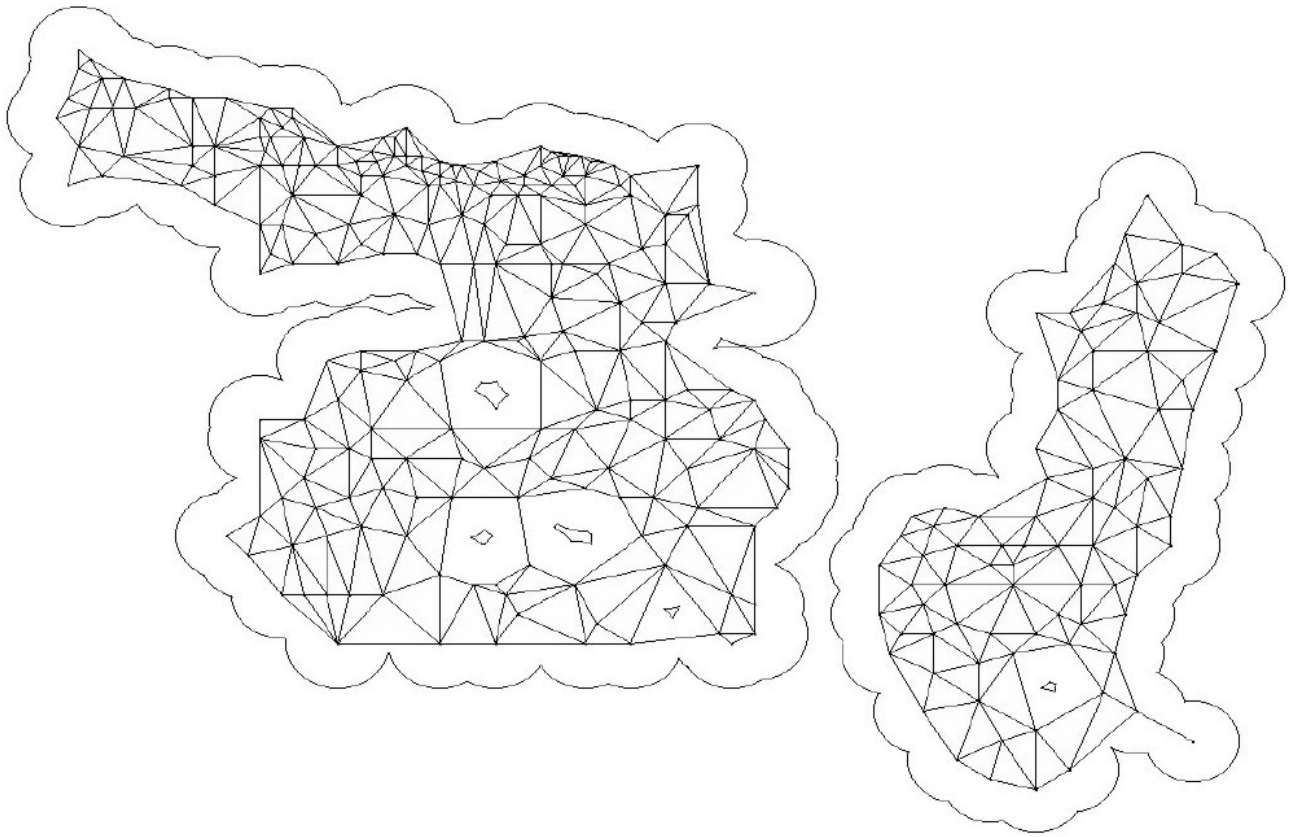


DELAUNAY–diagram



Alpha–shapes

Example



Alpha–shapes

short comings

- Need of suitable alpha.
- Alpha is constant over the entire point set.
- Interior points needed.

The algorithm is able to reconstruct

- a curve
- that is sampled sufficiently dense
- especially smooth curves, i.e., possibly many component curves without branches, endpoints, or self–intersections.

CRUST

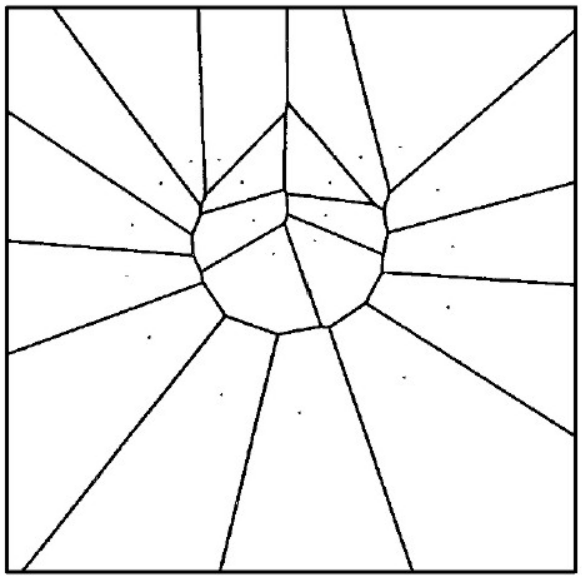
algorithm

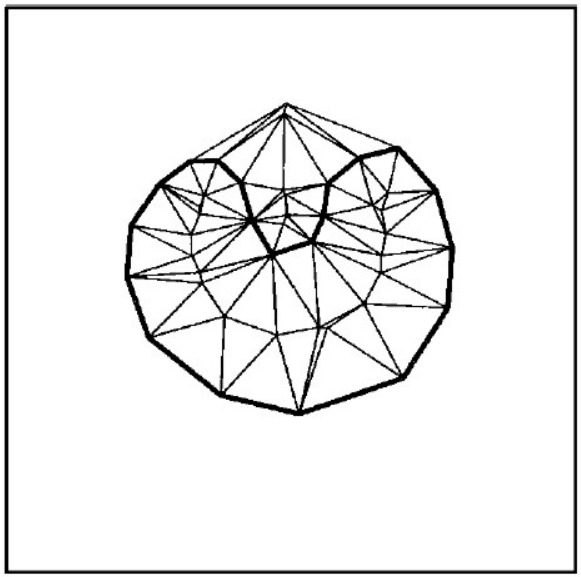
The crust

- is the set of edges
- selected from the Delaunay triangulation of the initial point set
- extended by its Voronoi points
- where both endpoints of the edges belong to the initial set.

CRUST

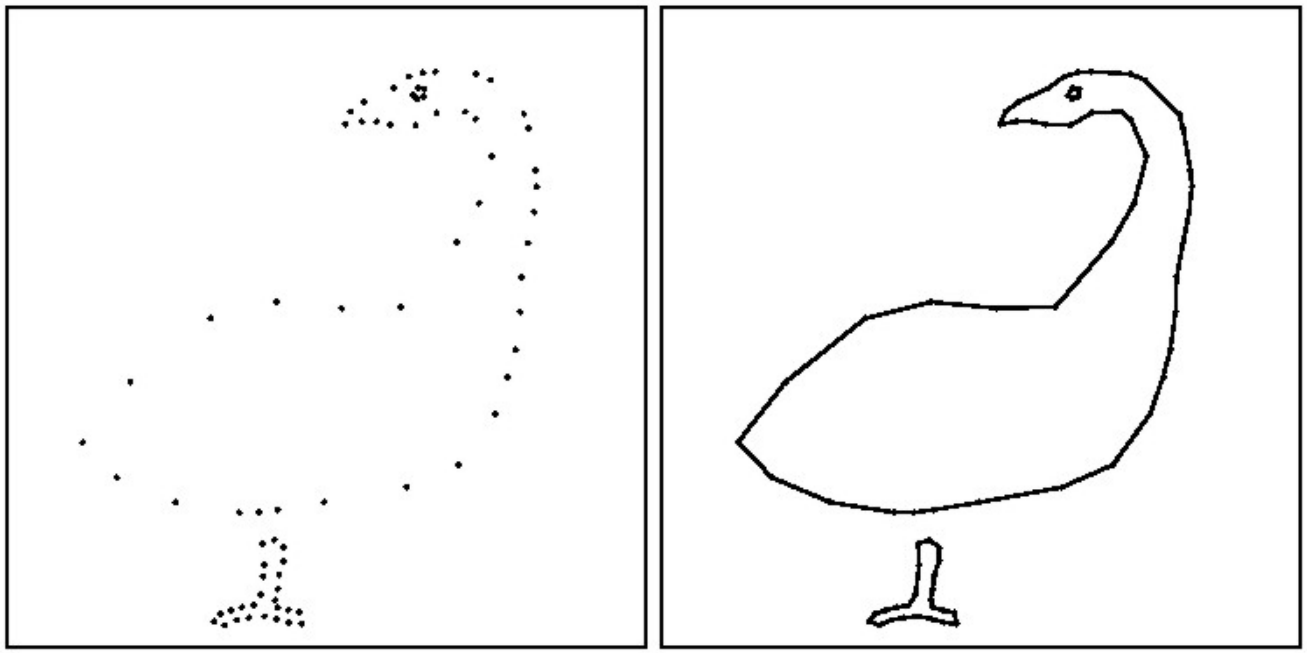
Example





CRUST

Example



Curve reconstruction

algorithm

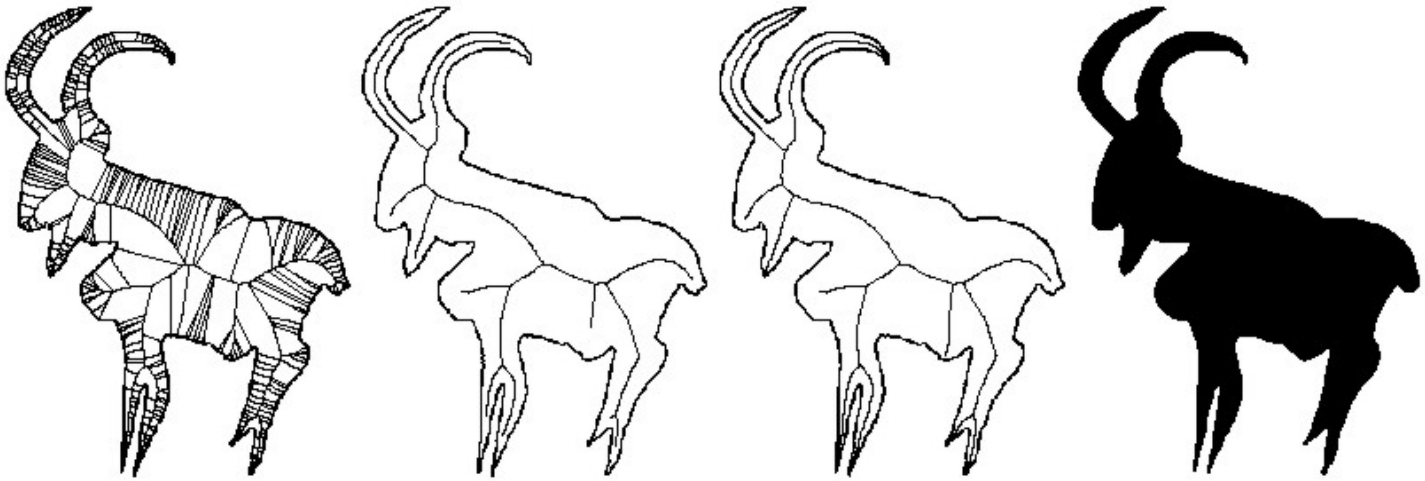
- Computes the GABRIEL graph as a subgraph of the Delaunay graph
- (an edge between two input points belongs to the Gabriel graph if a disk with this edge as diameter does not contain any other input point).
- Eliminate the edges which do not fulfill the local granularity property,
- i.e., at each point only the two shortest edges are maintained.

Curve reconstruction

application

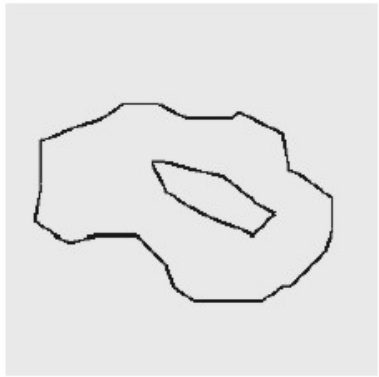
- The method works well if there exists a regular interpolant,
- that is, a polygonal closed curve such that the local granularity,
- defined as minimum distance to an input point,
- at each point of the curve is strictly smaller
- than the local thickness at that point,
- defined as the distance to the medial axis of the shape.

Skeleton

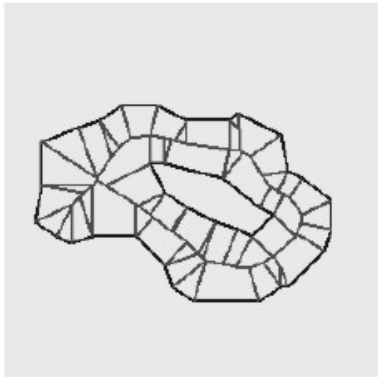


Skeleton

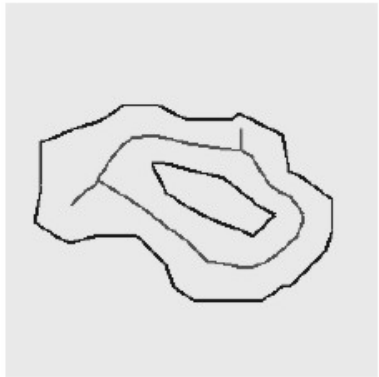
(a) Planar Shape



(b) Polygonal Approximation



(c) Voronoi Diagram



(d) Pruned Skeleton

Simplification problem

objective

- Given polygonal chain (or polygon) P (with n vertices),
- approximate P by another one Q whose vertices are a subset of k vertices of P.

Simplification problem

two variants

- min–#–problem: minimize the number of vertices of an approximating polygonal chain (or closed polygon) with the error within a given bound;
- min–ε–problem: minimize the error of an approximating polygonal chain (or closed polygon) consisting of a given number of vertices.

Simplification problem

solutions

- Both problems can be solved in optimal time $O(n^2)$.
- There exist near–optimal algorithms for solving the min–#–problem for the Euclidean distance which for practical problems outperform the optimal algorithms.
- There exists a genetic algorithm to cope with the min–#–problem which found near optimal solutions in the presented experiments.

Approximation problem

objective

- Given polygonal chain (or polygon) P (with n vertices),
- approximate P by another one Q whose k vertices can be placed arbitrarily in the plane.
- min–#–problem: minimize the number of vertices with the error within a given bound;
- min–ε–problem: minimize the error consisting of a given number of vertices.

Approximation problem

solutions

- There exists an algorithm that approximates the point set with a set of individual lines.
- Joining the lines is problematic in certain cases.
- There exists an algorithm that approximates with a linked chain whenever the input polyline is monotonic (runtime $O(n^k)$).
- There exists an approximation version of this algorithm which runs much faster (no implemention known).

A shape $\mathscr{S}$ is defined by a number of points and certain parameters:

- line: a point $L \in \mathbb{R}^2$ and an angle $\varphi \in \mathbb{R}$;
- circumference: a center point $C \in \mathbb{R}^2$ and a radius $\rho \in \mathbb{R}$;
- set of circumferences: set of pairs of center points and corresponding radii, i.e., $(\mathscr{C}, \mathscr{R}) \subset \mathbb{R}^2 \times \mathbb{R}$;
- polyline or polygon: an ordered set of corner points $\mathscr{Q} = \{Q_1, Q_2, \ldots, Q_k\}, Q_j \in \mathbb{R}^2, j = 1, \ldots, k$, where the only difference between the two shapes is that for a polygon the last and first corner are connected;
- rounded box: a line segment defined by two points $Q_1, Q_2 \in \mathbb{R}^2$, an aspect ratio $\alpha \in \mathbb{R}$, and a corner radius $\rho \in \mathbb{R}$.

- line

$$\delta_{L_2}(P_i, \mathscr{S}) = \left|\mathrm{Det}(P_i - L, (\cos\varphi, \sin\varphi)^T)\right|$$

- circumference

$$\delta_{L_2}(P_i, \mathscr{S}) = |\|P_i - C\| - \rho|$$

- set of circumferences

$$\delta_{L_2}(P_i, \mathscr{S}) = \min_{(C_j,\rho_j)\in(\mathscr{C},\mathscr{R})} |\|P_i - C_j\| - \rho_j|$$

- polyline or polygon, first we need the distance for a segment $\overline{QQ_j} = Q_{j+1} - Q_j$:

$$\delta_{L_2}(P_i, \overline{QQ_j}) = \begin{cases} \|P_i - Q_j\| & \text{if } (P_i - Q_j)^T \overline{QQ_j} < 0 \\ \|P_i - Q_{j+1}\| & \text{if } (P_i - Q_{j+1})^T \overline{QQ_j} > 0 \\ \left| \frac{\mathrm{Det}(\overline{QQ_j}, P_i - Q_j)}{\|\overline{QQ_j}\|} \right| & \text{otherwise} \end{cases}$$

and obtain for a polyline or polygon

$$\delta_{L_2}(P_i, \mathscr{S}) = \min_{Q_j \in \mathscr{Q}} \delta_{L_2}(P_i, \overline{QQ_j})$$

where for a polyline the index j runs from $1, \ldots, k-1$ and for a polygon from $1, \ldots, k$ with $Q_{k+1} = Q_1$.

- rounded box

$$\delta_{L_2}(P_i, \mathscr{S}) = \left| \min_{j=1,\ldots,4} \delta_{L_2}(P_i, \overline{QQ_j}) \pm \rho \right|$$

where $Q_3 = Q_2 + \alpha\, Q_{12}^{\top}$ and $Q_4 = Q_1 + \alpha\, Q_{12}^{\top}$ being $Q_{12}^{\top}$ the left turned perpendicular vector to $Q_2 - Q_1$ of same length. $+\rho$ is taken when the point P_i lies inside the rectangular box through $Q_1, \ldots, Q_4$, and $-\rho$ when P_i lies outside.

- vertical distance to a line:

$$\delta_V(P_i, \mathscr{S}) = \left| P_i^2 - L^2 - (P_i^1 - L^1) \cdot \sin\varphi / \cos\varphi \right|$$

- length of the segments could have an influence as a weight

$$\delta_{wL_2}(P_i, \overline{QQ_j}) = \left| \mathrm{Det}(\overline{QQ_j}, P_i - Q_j) \right|$$

- RMS

$$\delta_{\mathrm{RMS},L_2}(\mathscr{P},\mathscr{S}) = \sqrt{\frac{1}{l}\sum_{P_i\in\mathscr{P}} \delta_{L_2}(P_i,\mathscr{S})^2}$$

- AVG

$$\delta_{\mathrm{AVG},L_2}(\mathscr{P},\mathscr{S}) = \frac{1}{l}\sum_{P_i\in\mathscr{P}} \delta_{L_2}(P_i,\mathscr{S})$$

- MAX

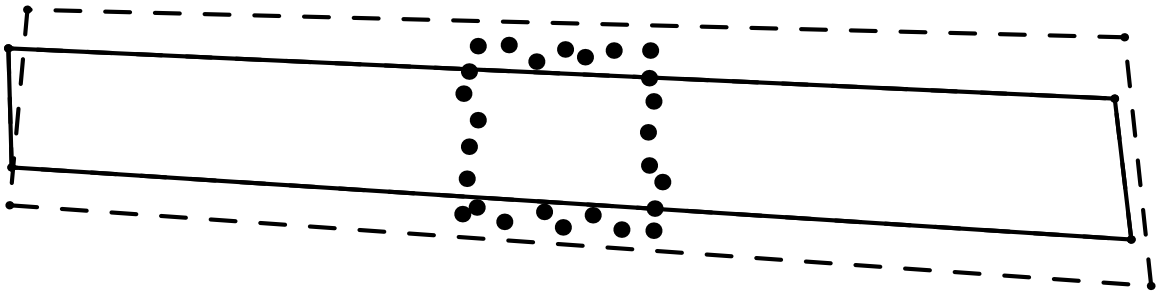
$$\delta_{\mathrm{MAX},L_2}(\mathscr{P},\mathscr{S}) = \max_{P_i\in\mathscr{P}} \delta_{L_2}(P_i,\mathscr{S})$$

- or $\delta_{\mathrm{RMS},V}(\mathscr{P},\mathscr{S})$ or $\delta_{\mathrm{AVG},L_1}(\mathscr{P},\mathscr{S})$, etc

- Algorithm of RODRÍGUEZ/GARCÍA–PALOMARES
- derivative–free minimization method
- proved convergence for either locally strictly differentiable or non–smooth locally convex functions.

- Til now concentrated on the minimization of a single function.
- However, in our optimization problem, it can happen that there is no change in the value of the distance function although the shape is modified.

Example: Subsidiary objective



Subsidiary objective: idea

- Minimize perimeter as well.

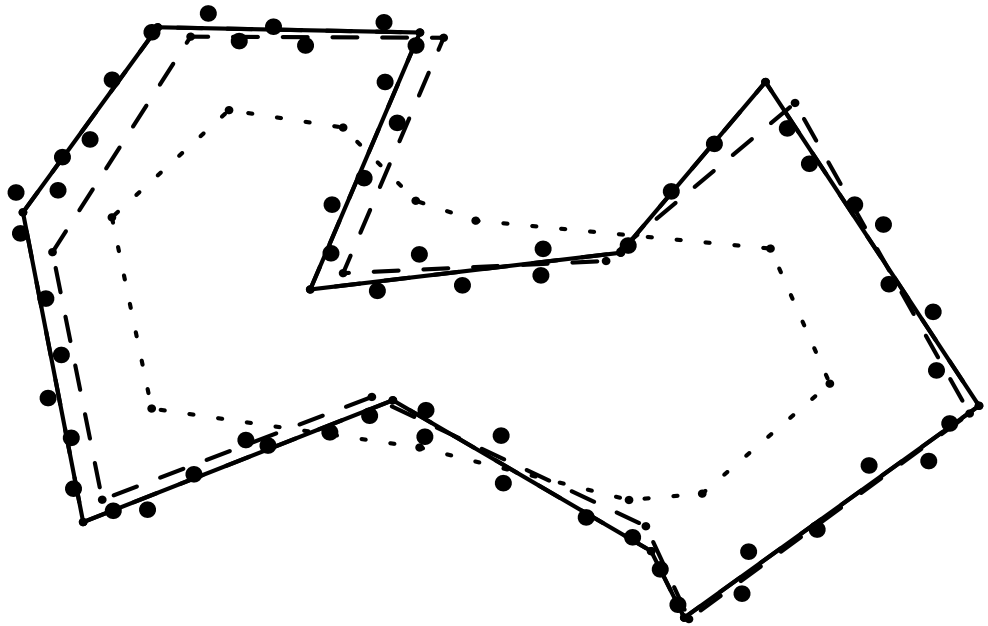
- Formulate a convex combination of all objectives,
- i.e., optimize the single–objective function

$$f(x) = \beta_1 f_1(x) + \beta_2 f_2(x) + \cdots + \beta_l f_l(x)$$

- where the $\beta_j > 0$, for $j = 1, \ldots, l$, are strictly positive weights,
- $f_j(\cdot) : \mathbb{R}^n \longrightarrow \mathbb{R}$ are the individual objective functions.

- Explores the Pareto front defined by the weights β_j,
- it might be difficult to find weights such that the encountered minimum is sufficiently close to the minimum considering only the principal objective $f_1(\cdot)$.

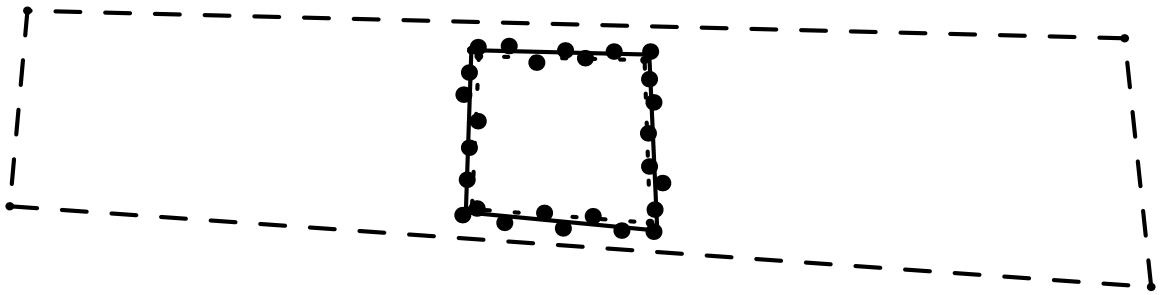
Example: convex combination as objective function



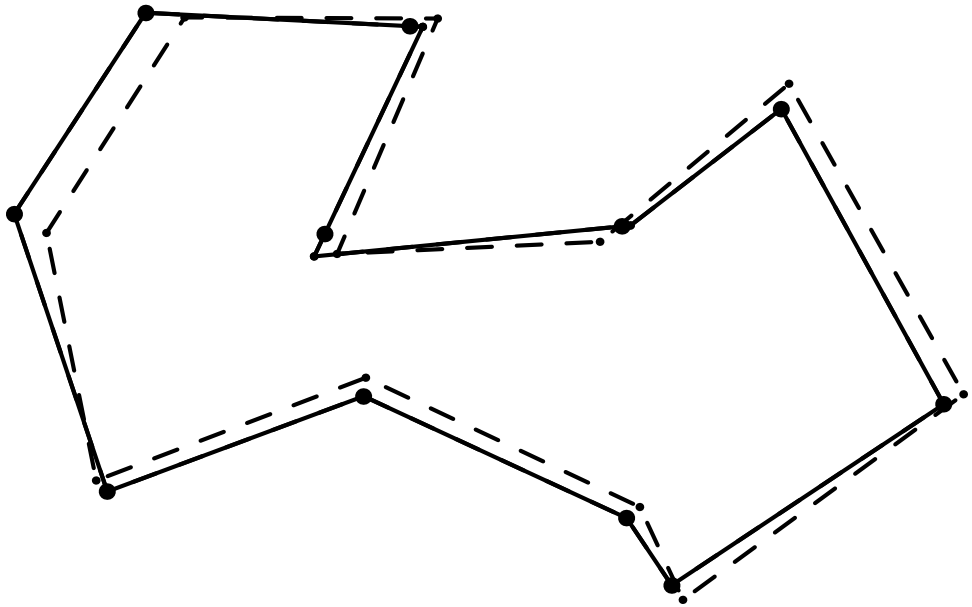
- modify the comparison $f(z) \leq f(x) - \tau^2$ (in the local search algorithm)
- with the following iteratively defined comparison function for l objectives:

$$[f_1(z), \ldots, f_l(z)] \leq_{\tau^2} [f_1(x), \ldots, f_l(x)] \Longleftrightarrow$$
$$\forall j = 1, \ldots, l : f_j(z) \leq f_j(x) - \tau^2 \text{ or}$$
$$(j < l \text{ and } f_j(z) \leq f_j(x) \text{ and } f_{j+1}(z) \leq f_{j+1}(x) - \tau^2)$$

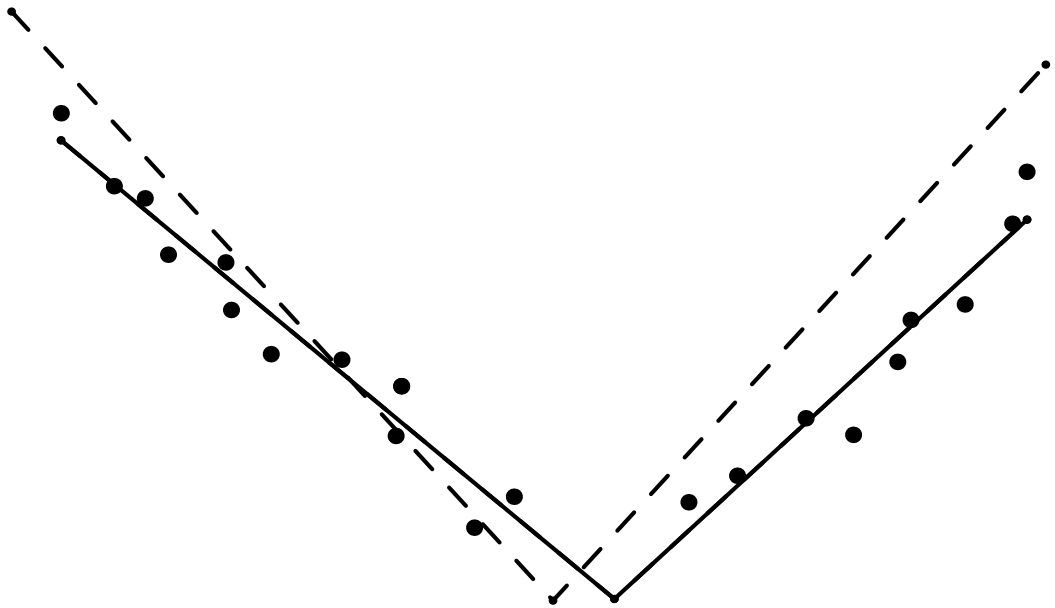
Example



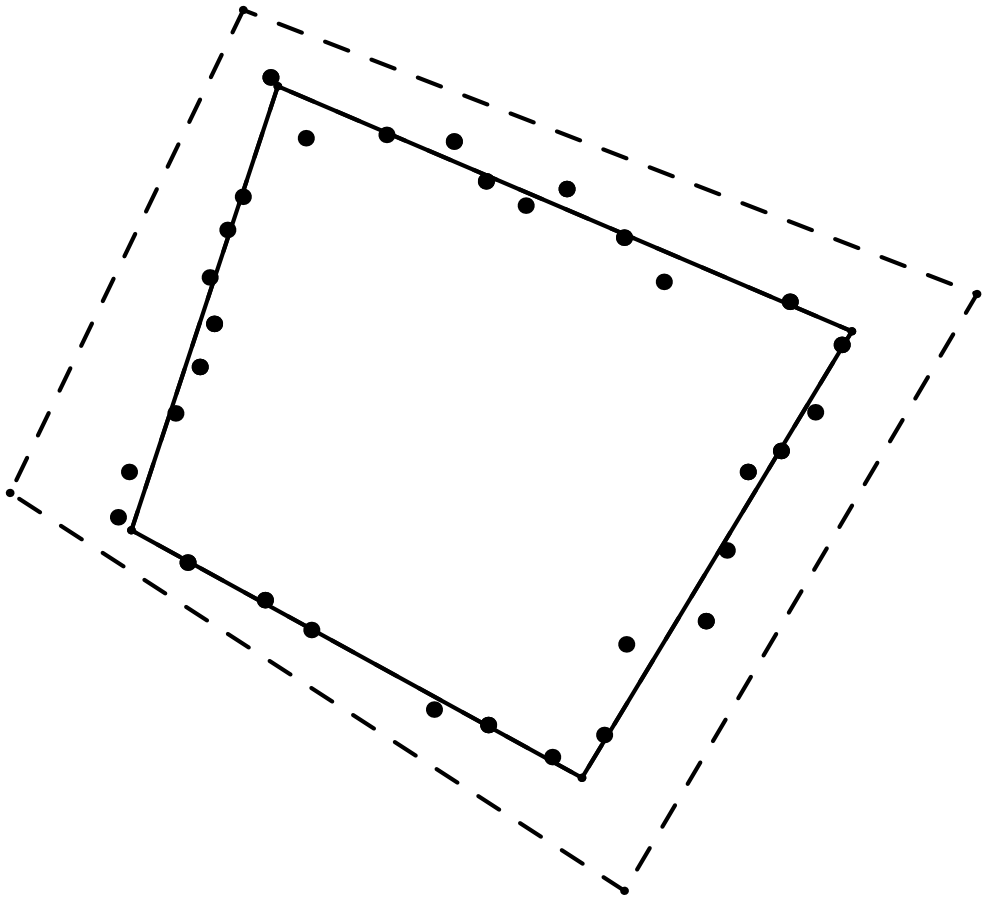
NP–Completeness and non–convexity of the objective function



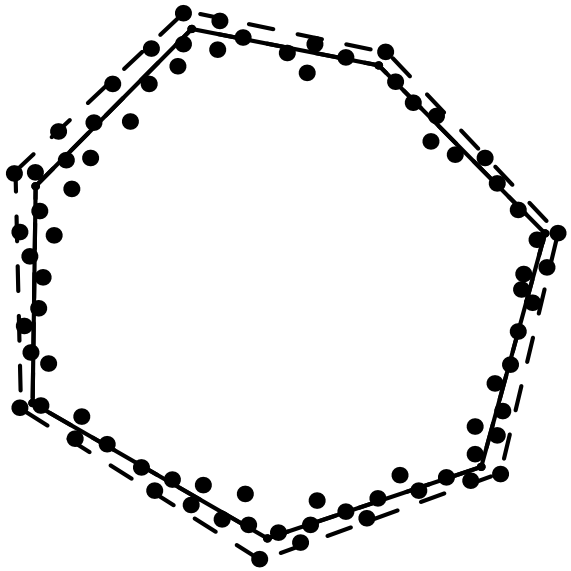
Examples: wedge



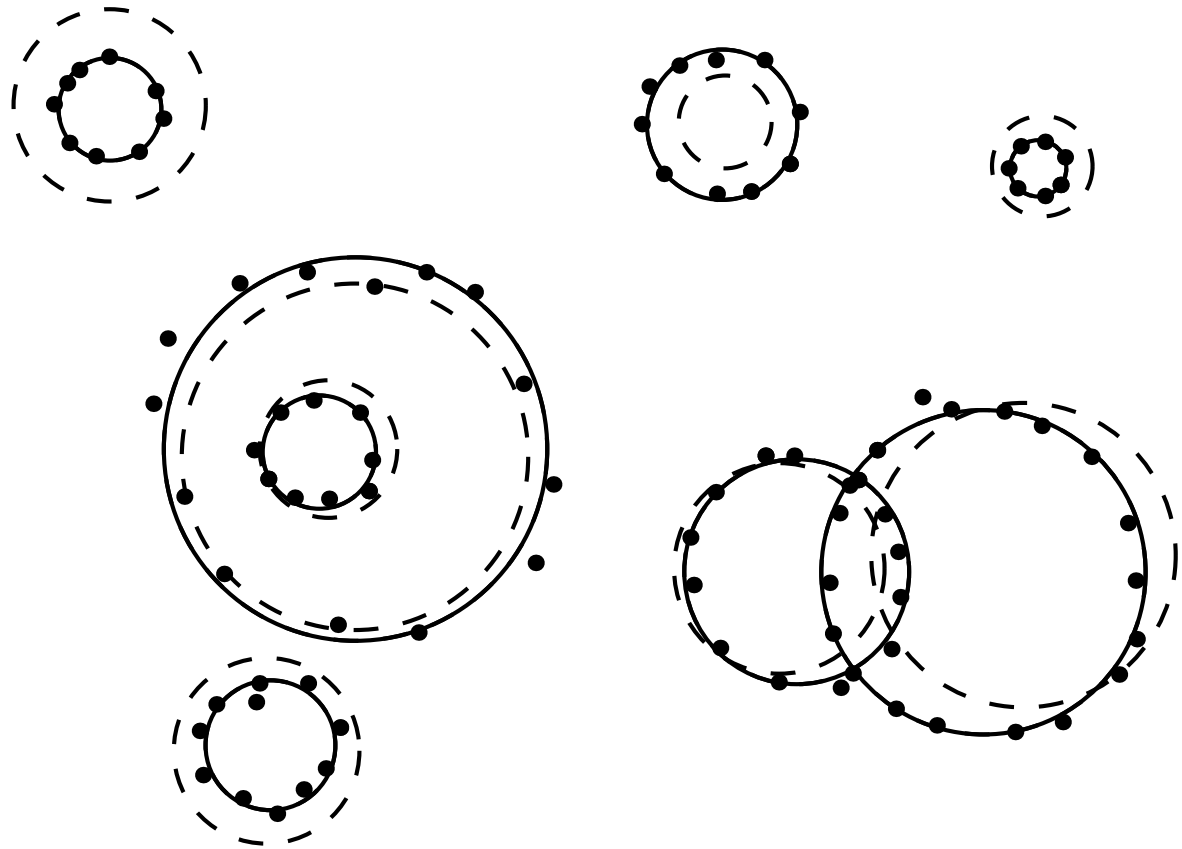
Examples: simple polygon



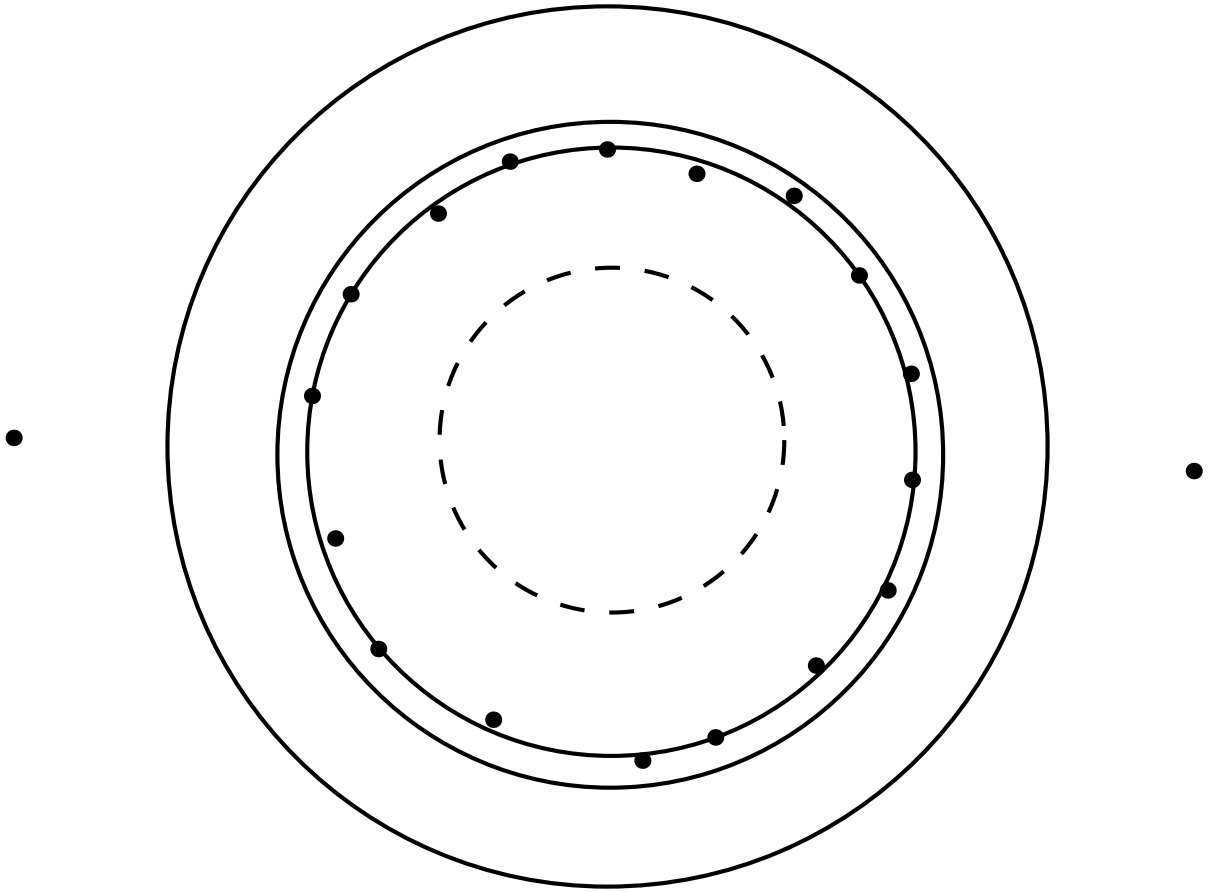
Examples: convex hull



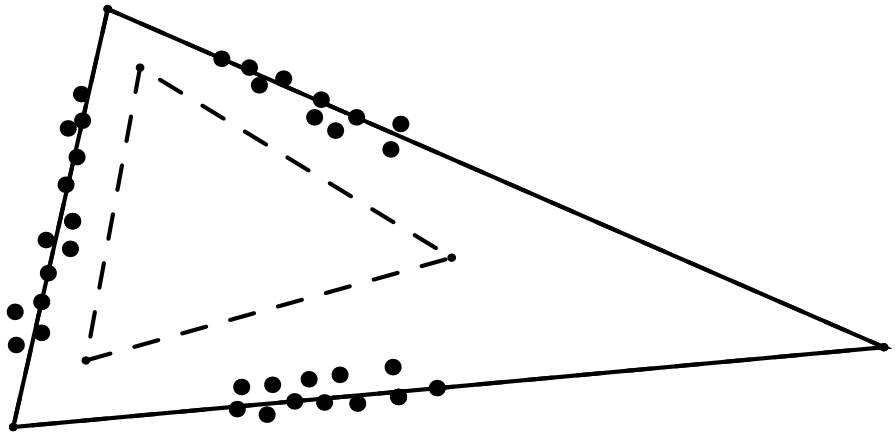
Examples: circumferences



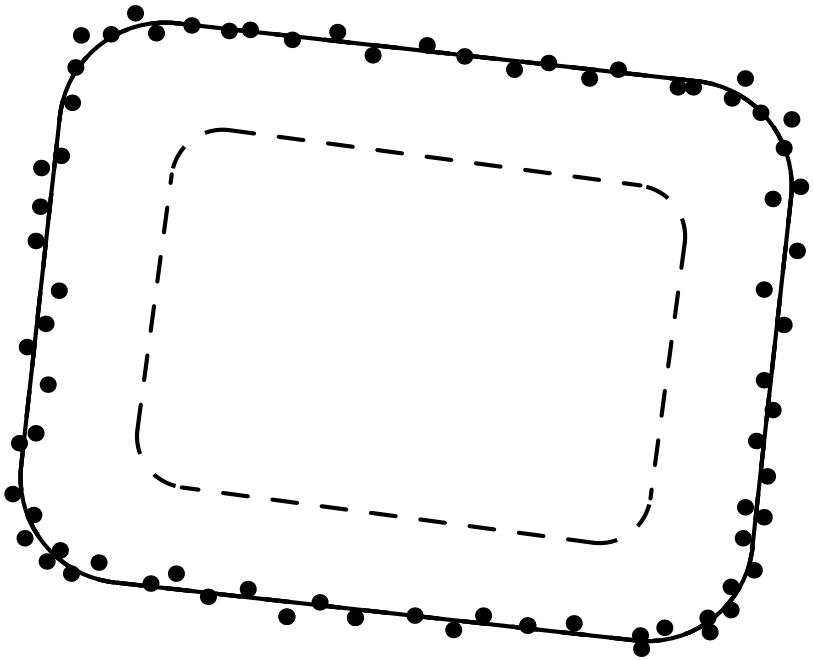
Examples: influence of metrics



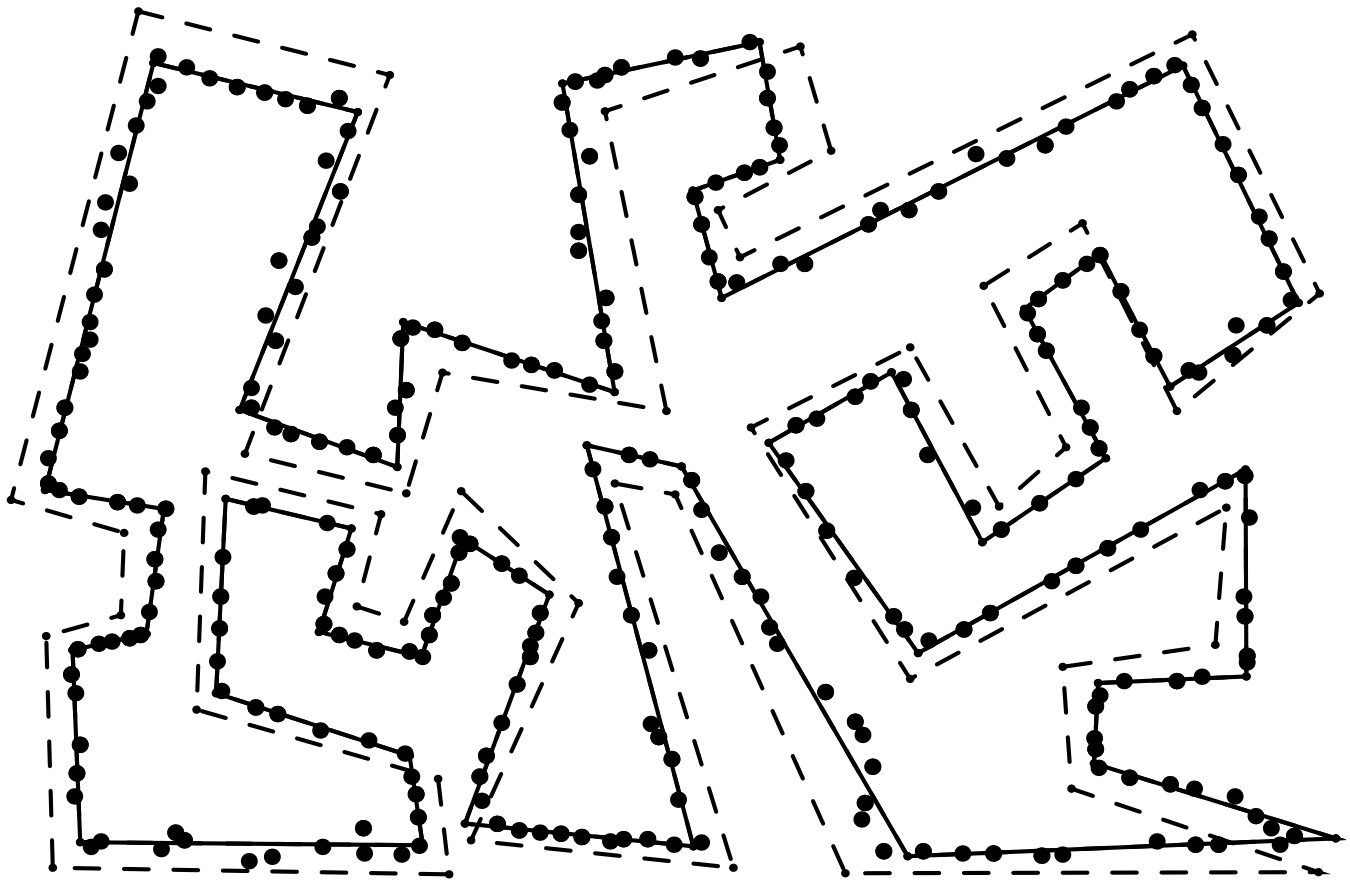
Examples: completing shapes



Examples: rounded rectangle



Examples: complex polygon



Evolutive methods

based on natural fenomena

- simulated anealing
- cristalizaton of materials
- evolution (mutation, recombination, selection)
- competitive/colaborative systems
- social interactions

Evolutive methods

with meta heuristics

- (reactive) tabú search (since 1986)
- random search (since 196X)
- simulated anealing (since 196X)

- genetic algorithms (since 1975)
- genetic programming
- (neural networks)
- ant colony optimization (since 1992)
- particle swarm optimization (since 1995)
- guided local search (since 1997)
- iterated local search (since 1999)
- variable neighborhood search (since 1999)

Evolutive methods

paradigms

- work with populations of individuals
  (only one individual and a memory...)
- there are modification processes
  (mutation, modification, reproduction)
- *performance* of the individuals in the environment based on a *fitness* which usually is the objective function (but not necessarily)
- decisions are drawn probabilistically

Evolutive methods

genetic algorithms

- one distinguishes genotype (codification) and fenotype
- there exists a bijection between genotype and fenotype
- modifications (mutation and crossover) is done over the genotypes
- the fitness is evaluated over the fenotypes
- mutation (which one?), recombination (types?), selection (types?)

Evolutive methods

evolutive programming

- there exists only the fenotype (with its codification)
- modification (mutation) is realized over the fenotypes of copies
- the fitness is evaluated over the fenotypes
- mutation (types?), selection (types?)

Evolutive methods

evolutive strategies

- an amplification of evolutive programming
- each individual maintains parameters that guide the mutations
- these parameters are modified in the same way as the proper fenotypes
- mutation (types?), selection (types?)

Evolutive methods

genetic programming

- the codification of the fenotype is a program
- the programs are modified with adecuate operations
- mutation (types?), selection (types?)

recent example (André Falcão, Residue fragment programs for enzyme classification, Proceedings BKDB2005, pp.24–28, 2005).

Evolutive methods

differential evolution

- the codification of the fenotype is a vector of characteristics
- the vector of an individual is modified with the differences to other vectors (individuals)
- modifications (types?), selection (types?)

Evolutive methods

swarm intelligence

- the individuals of the population interact in a social way
- the decisions of each individual depend on the own wishes and the information available from the others
- ant colonies
- particle swarms

Evolutive methods

ant colonies

- the individuals leave information (feromonas) in the search space
- the decisions are based on individual information and on the feromonas encountered
- the information (feromonas) is volatile
- the feromonas or statistical behavior of the individuals define the solution

Particle swarm optimization

characteristics

- simple to describe
- simple to implement
- few parameters to adjust
- usually small population are used
- the number of objective function evaluations is usually small
- usually is very fast

premature convergence occurs whenever all individuals are located in a small area of the search space

Particle swarm optimization

some details

- each individual communicates with its neighborhood
  (usually, the neighborhoods overlap)
- and maintains local information
  (best solutions viewn til now, search direction, etc.)
- in most cases, the neighborhood is fixed
- the local information is modified with the help of the information gathered in the neighborhood
  (or just from the best neighbor)
- local changes are confined to avoid *explosions*
  (dramatic changes)
- the method is able to solve discrete problems

Particle swarm optimization

velocity actualization

$$
\begin{aligned}
v_i &= \xi(v_i + U[0,\varphi_1](p_i - x_i) + U[0,\varphi_2](p_g - x_i)) \\
x_i &= x_i + v_i
\end{aligned}
$$

con

- x_i vector of current positions
- v_i vector of current directions
- p_i best local position vector
- p_g best position vector of group (neighborhood)
- $\varphi_1 = 2{,}05$
- $\varphi_2 = 2{,}05$
- $\xi = 0{,}729$

Particle swarm optimization

versions

binary version: the variables are interpreted as binary values according to a distribution of threshold

discret version: the variables are interpreted as integer values (for instance with simple rounding)

dynamic version: the search space is reinitialized, the local variables are reset, for instance: $p_i = x_i$ or re-evaluate p_i and decide between p_i ans x_i.

Particle swarm optimization

convergence

- the individuals should exhibit certain diversity
- one needs a similarity measure
- diversity can be forced dynamically adapting the parameters
- one might use lack of diversity as stopping condition

multi–objective optimization

Pareto

- there is more than one independent objective function
- Pareto optimal (global): every other component for all other solutions is worse (or equal)
  (other names are: efficient points, dominant points, non-interior points)
- Pareto optimal (local): every other component for all other solutions in a local neighborhood is worse (or equal)
- the Pareto frontier describes the trade-off between the different objectives

Multi–objective

techniques for a solution

- convex combination of the objectives
  (to obtain the Pareto frontier one has to explore the coefficient space)
- homotopic techniques, i.e., compute the entire Pareto frontier (works in most cases just for two objectives)
- goal programming, i.e., fix values for all objectives and minimize the distance of all objectives to the predefined goals (according to some convenient distance metric)
- priority optimization, i.e., fix thresholds for all but one objective beforehand and optimize above the threshold according the most important one
- priorization (multi-level) programming, i.e., optimize according to a predefined ordering of the objectives.

Multi–objective optimization

with evolutionary methods

- evolutionary methods can approximate the Pareto frontier in parallel (with the help of the diversity among the individuals)
- for instance particle swarm systems varying the weights of a convex combination periodically during the iterations