Intelligent and Adaptable Software Systems

Advanced Algorithms: Optimization and Search Methods

Dr. Arno Formella

Computer Science Department
University of Vigo

09/09

Advanced Algorithms: Optimization and Search Methods I

Course organization

course notes

- Homepage:
  `http://www.ei.uvigo.es/~formella/doc/ssia09`
- whiteboard illustrations (notations, ideas for proofs, algorithms)
- very short introduction to certain aspects related to optimization and search methods and some applications

Course organization

class room hours

Optimization and Search Methods, Thursdays, 18:00–20:00

24.09. class	01.10. class	08.10. lab	15.10. class	22.10. lab
29.10. class	05.11. ??	12.11. ??	19.11. ??	26.11. ??
03.12. ??	10.12. ??	17.12. ??	07.01. ??	14.01. ??

Course organization

class room hours

- Dr. Fernando Diáz Gómez
  office hours: `fdiaz@infor.uva.es`
- Dr. Arno Formella
  office hours: Mondays, 11-14 and 17-20

Bibliography

books

- to be completed

Your work

homework, lab hours, presentations

- to be completed

Bibliography I

links

(working in september 2009)

- Rui Mendes. *Population topologies and their influence in particle swarm performance*. PhD Thesis, Universidad de Minho, 2004.
  `http://www.di.uminho.pt/~rcm/`
- `http://www-neos.mcs.anl.gov`
  Online optimization project
- `http://www.coin-or.org/index.html`
  Operation research
- `http://www.cs.sandia.gov/opt/survey`
  global optimization

Bibliography II

links

- `http://iridia.ulb.ac.be/~mdorigo/ACO/`
  Ant colony optimization
- `http://www.mat.univie.ac.at/~neum/glopt.html`
  Global optimization
- `http://plato.asu.edu/gom.html`
  Continuous global optimization software
- `http://www.swarmintelligence.org/index.php`
  Particle swarm optimization

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

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
- differentiable or non–differentiable
- static or dynamic

The objective function must be confined.

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

Basic concepts

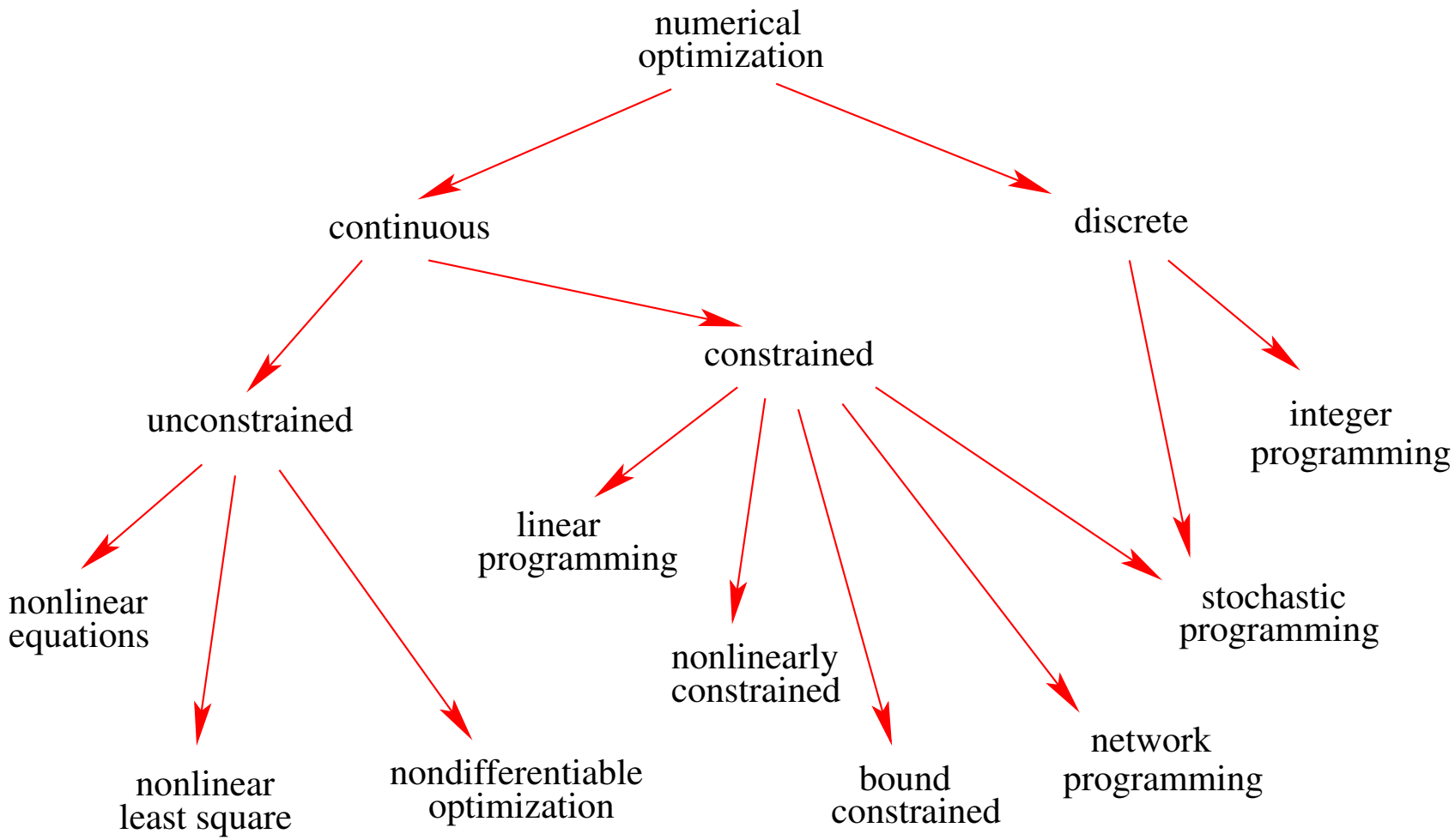
conditions

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

Basic concepts

classification

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



to be distinguished

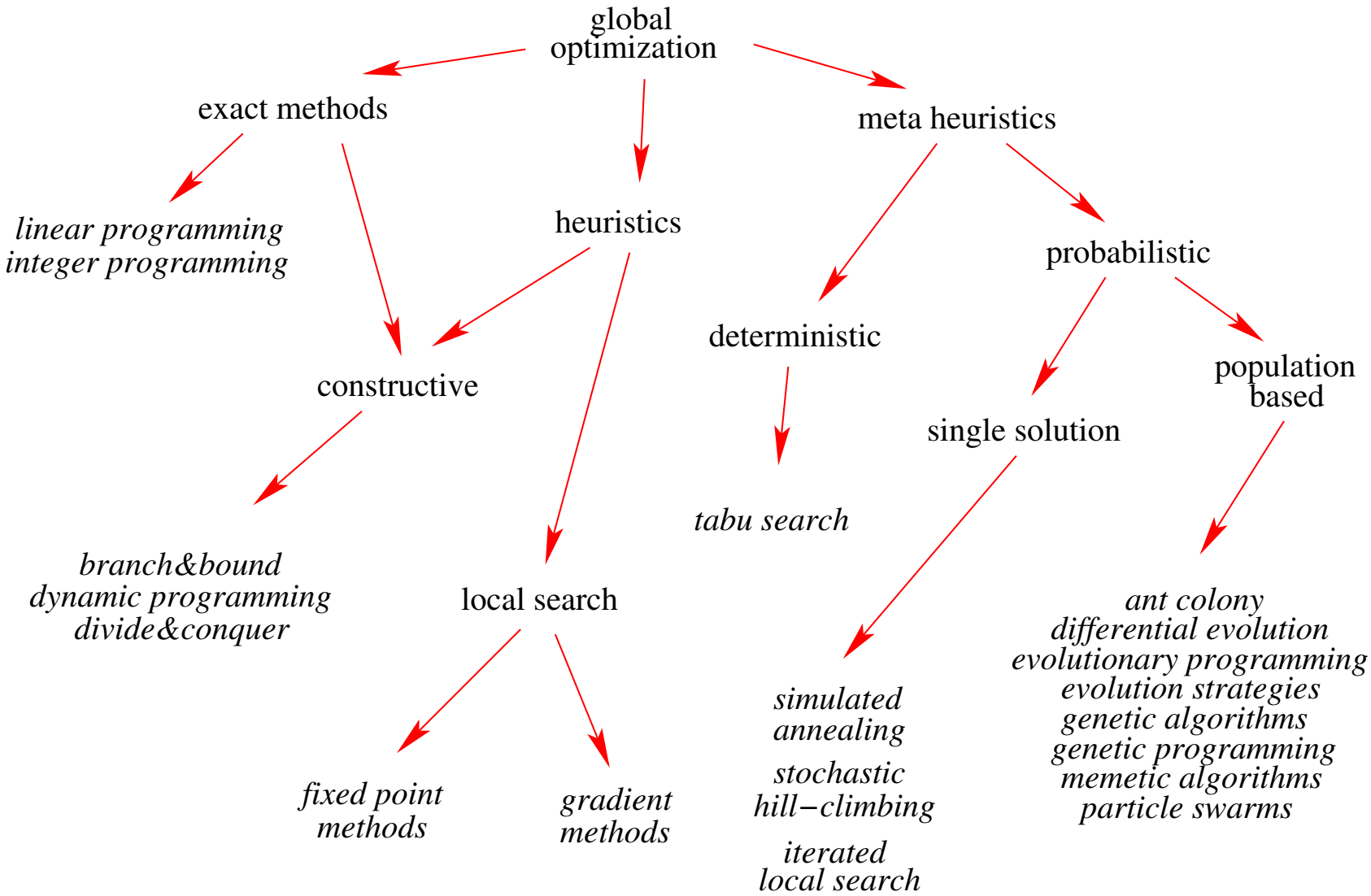
local optimization: usually one starts from an initial solution and stops when having found a local (close) minimum

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

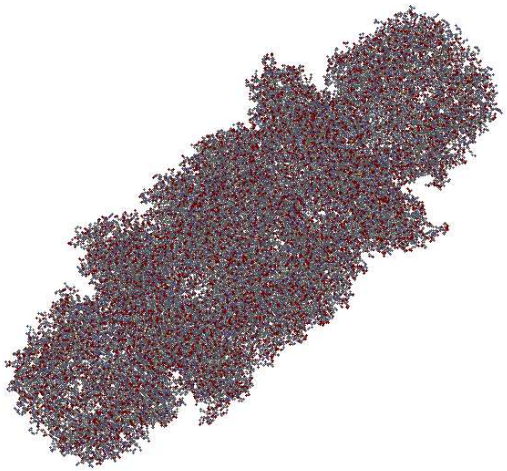
- searching of patterns
  (relatively small sets of two– or three–dimensional points),
  within search spaces
  (relatively large point sets)
- comparing point sets
- key words
  geometric pattern matching, *structure comparison*, *point set matching*, *structural alignment*, *object recognition*

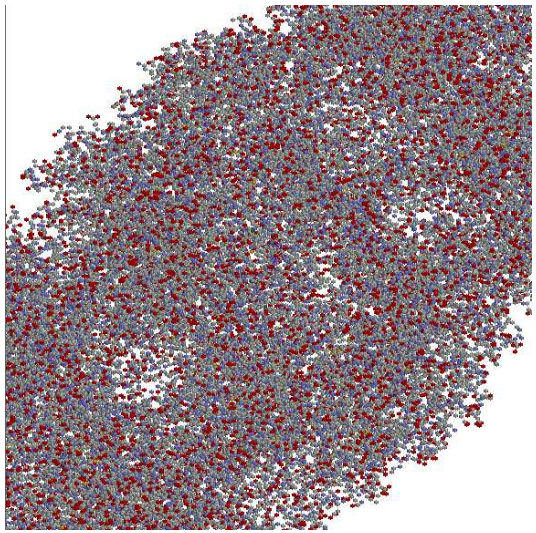
Joint work with

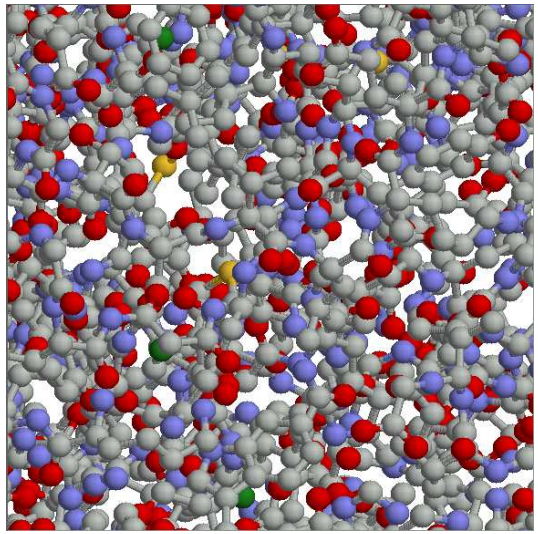
- Thorsten Pöschel
- some ideas from: Kristian Rother, Stefan Günther
- Humboldt Universität—Charité Berlin
  `http://www.charite.de/bioinf/people.html`
- `psm` is one of the algorithms available at
  `http://farnsworth.charite.de/superimpose-web`

Search of a substructure in a protein

search space

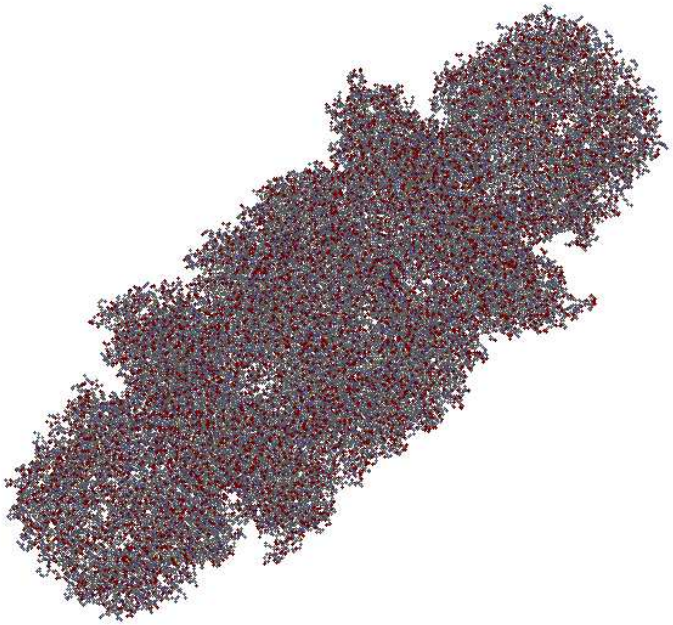


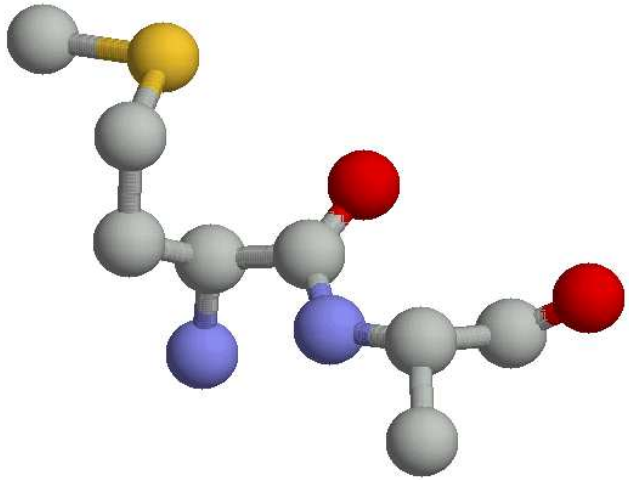




Search of a substructure in a protein

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)

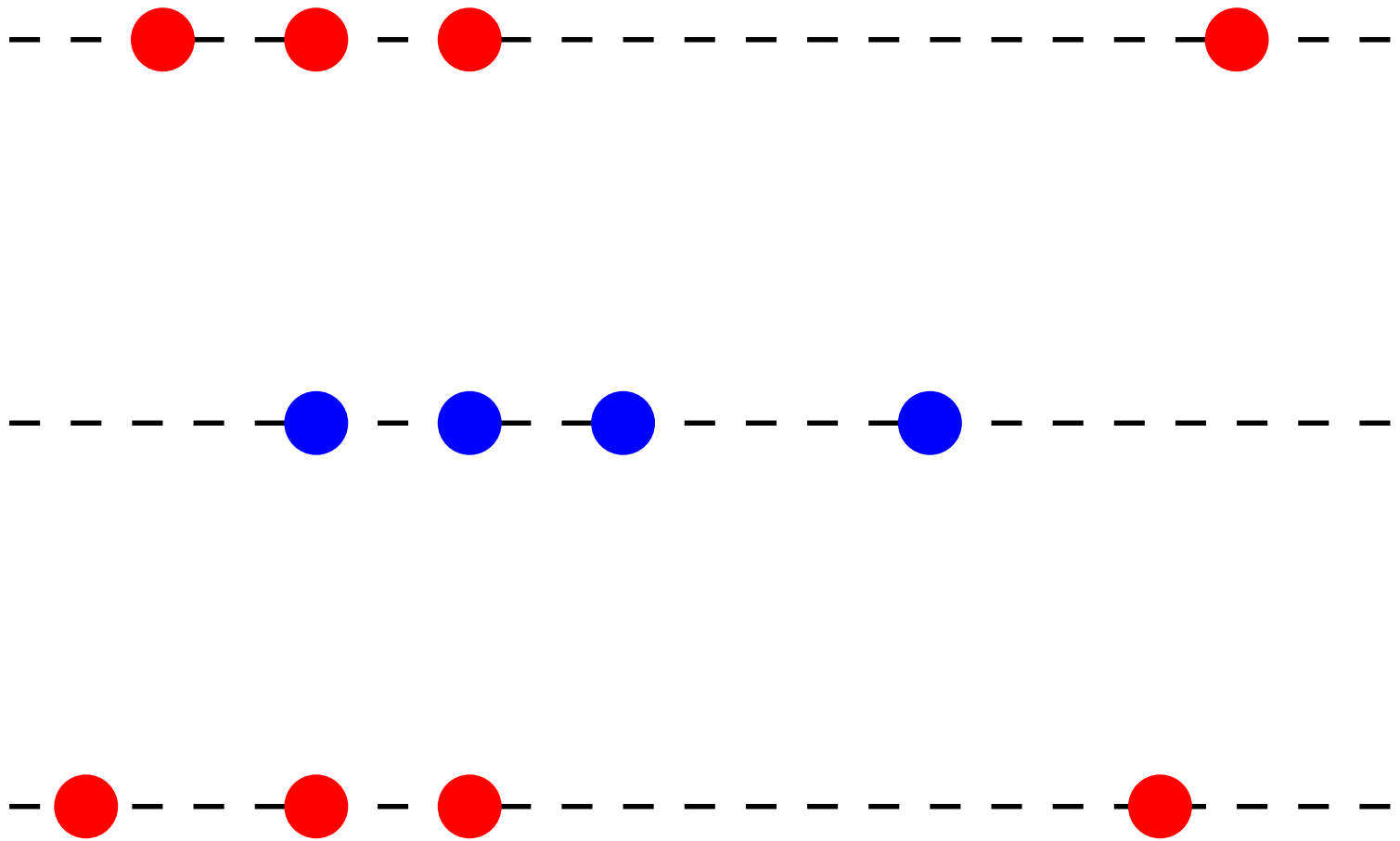
Congruent and similar transformations

- 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

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

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 $U = \mathbf{B} \cdot \mathbf{A}^\top$

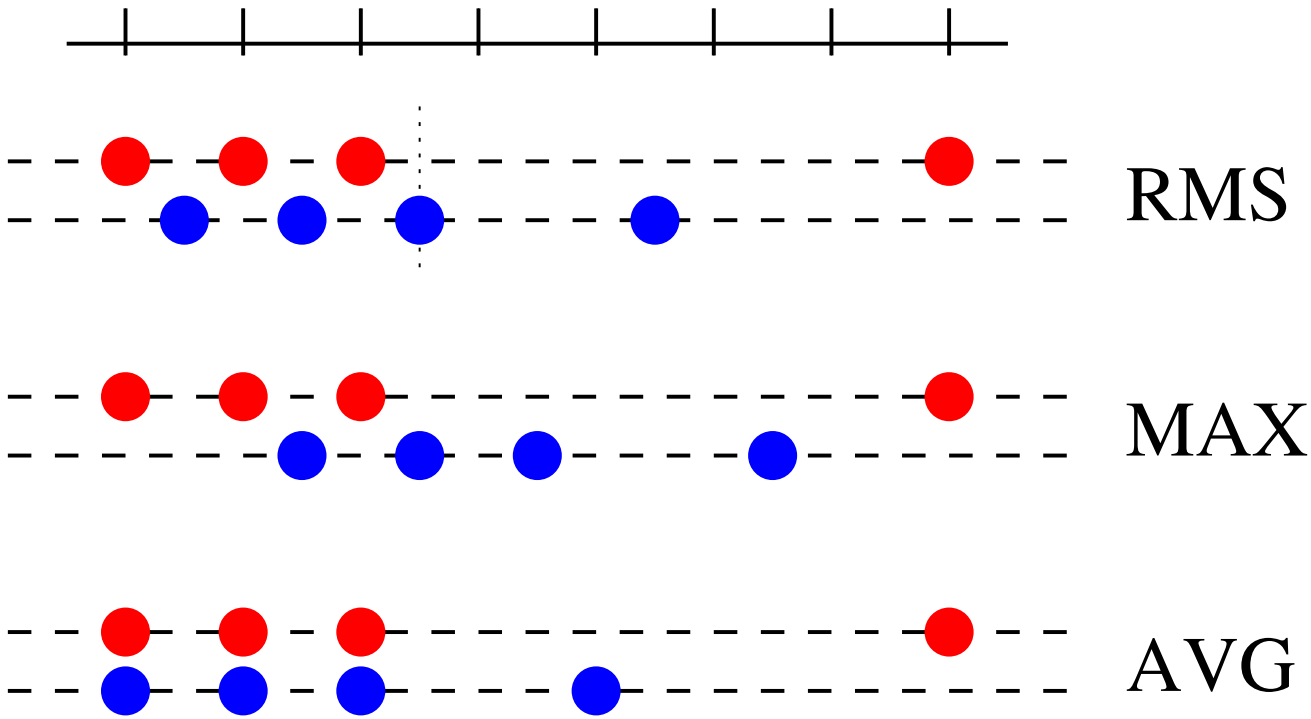
Introduction of scaling

- 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

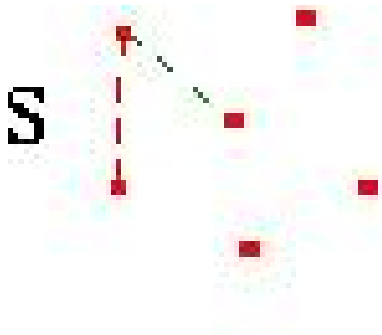
Properties of Clique Detection

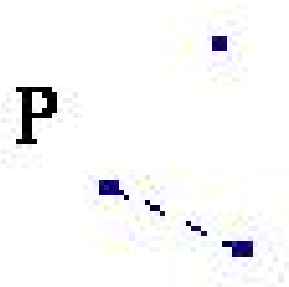
- the problem is NP–complete
  (however, we search only for *cliques* of size $\leq k$)
- fast algorithms need adjacency matrices
- if $n = |S| = 5000$ and $k = |P| = 100$ we need 30 GByte
  (counting only one bit per edge)

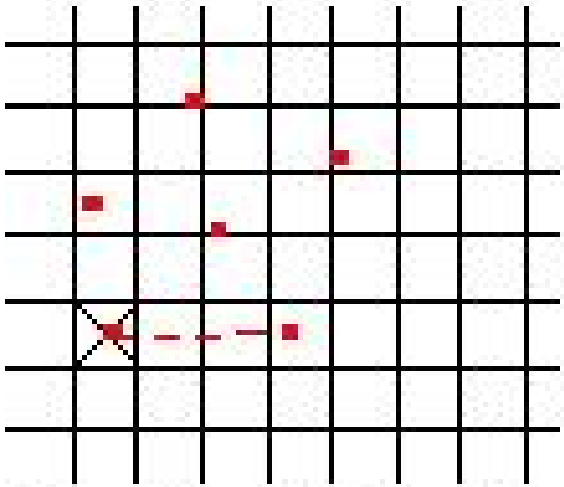
Geometric hashing

- 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

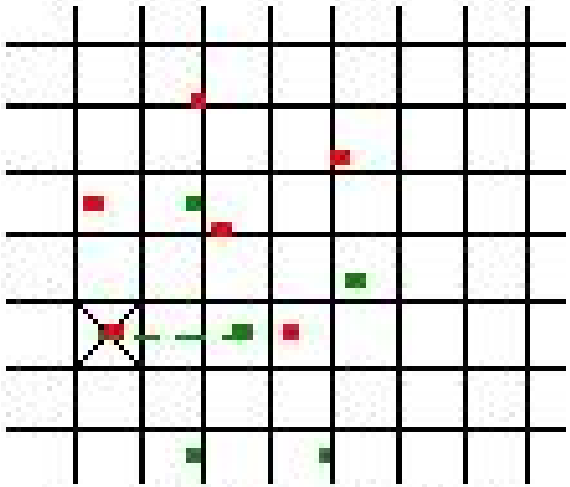
Example: geometric hashing







first insertion



second insertion

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

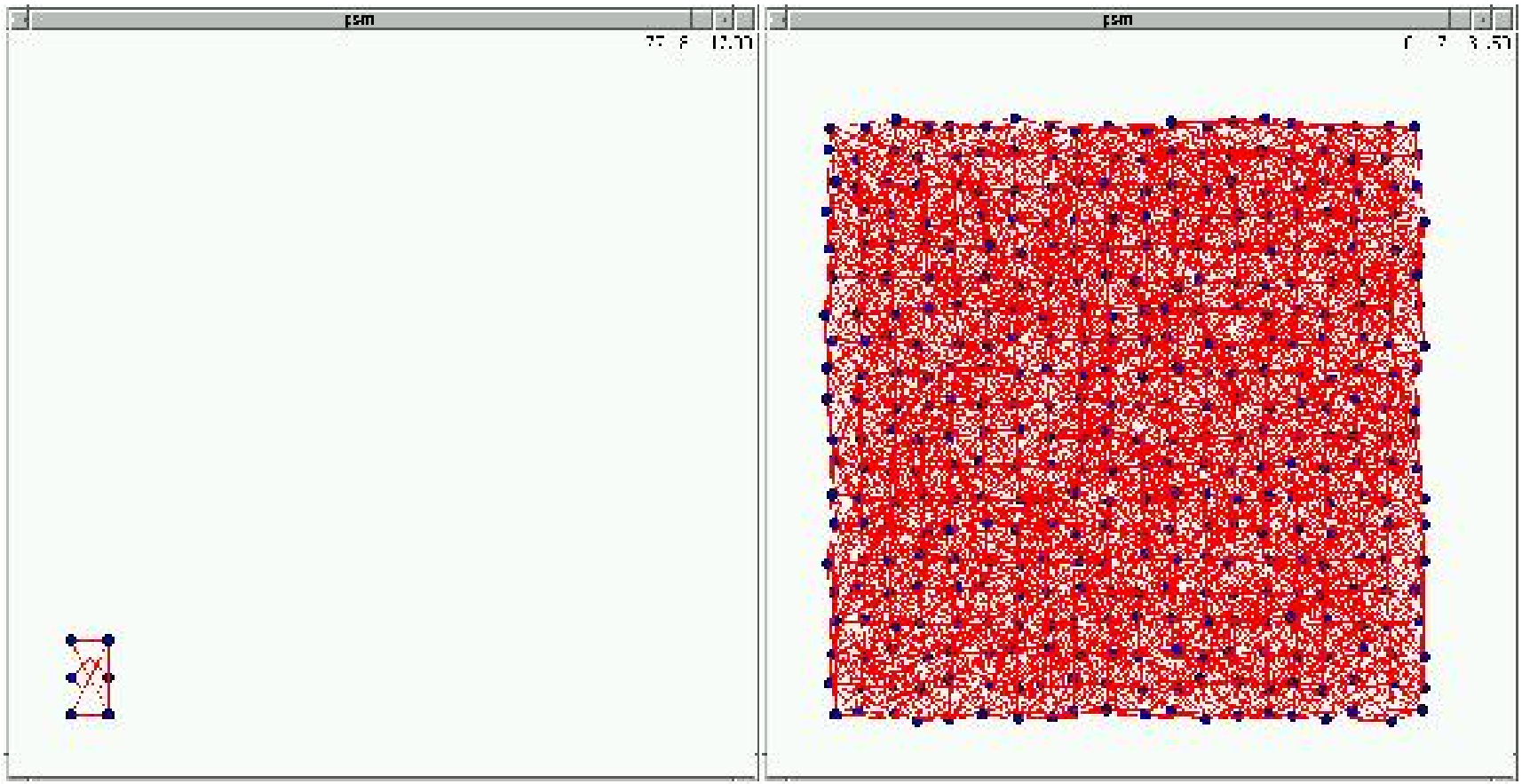
Construction of the dictionary

- 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

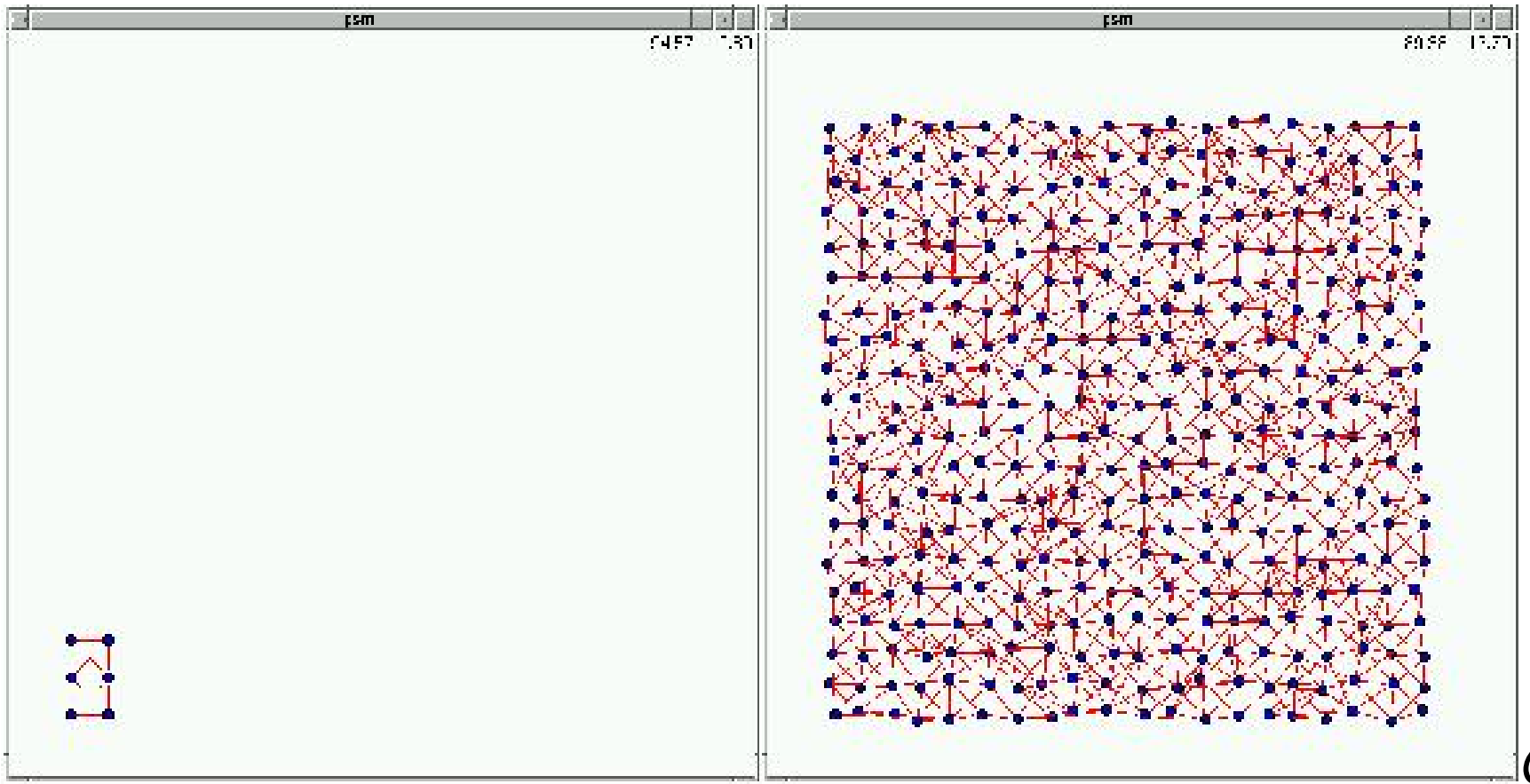


G_P G_S

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 $|\mathcal{P}(P)| = 2^k$ 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 si 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: $|P| + 1$

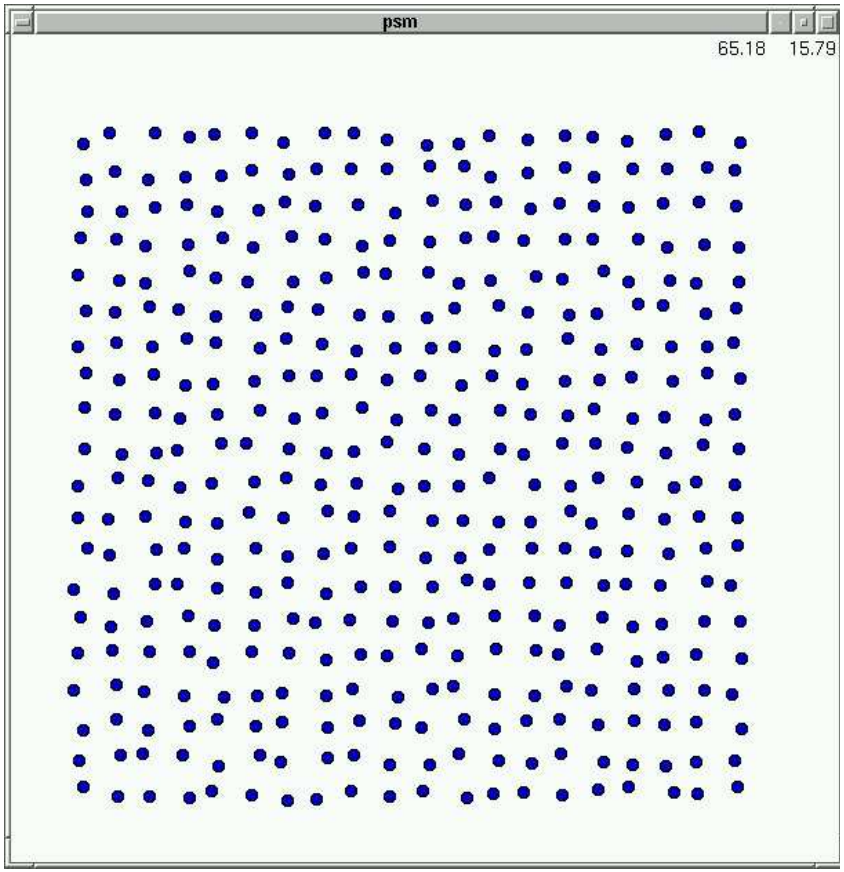
Reactive Tabu Search for psm

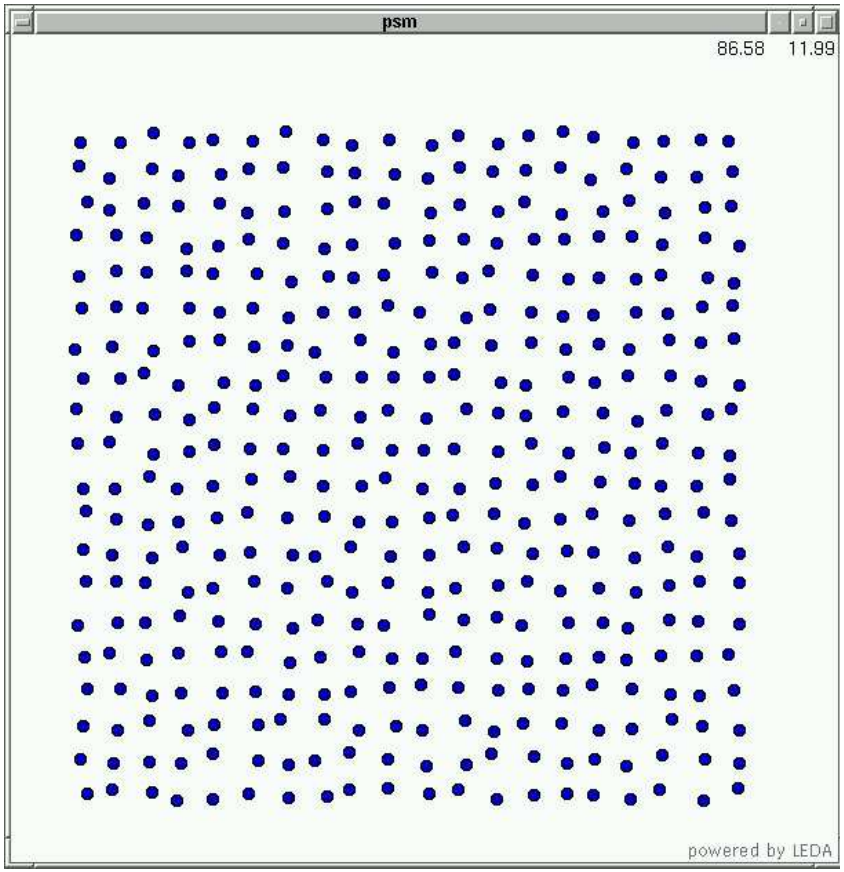
- representation of the problem:
  sets of indices of the matched points
- search for candidates to improve (*add*):
  (rigidly) connected neighbors within graph G_S
- search for candidates to reduce (*drop*):
  any point of the current solution that mantains the graph G_S connected (and rigid)

¿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

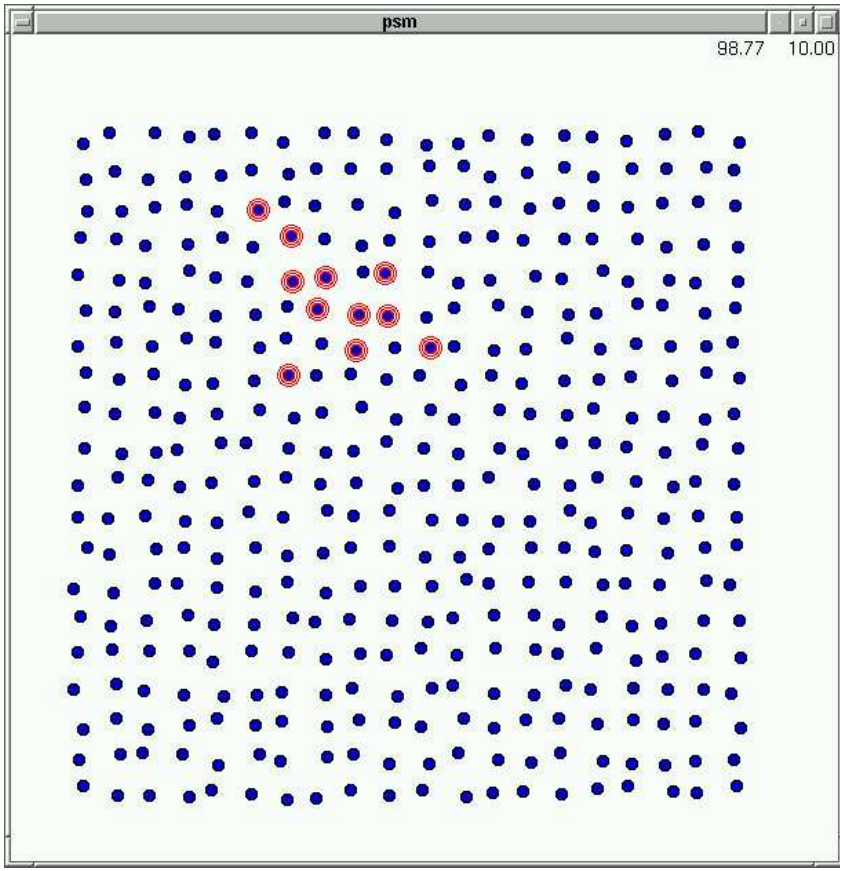


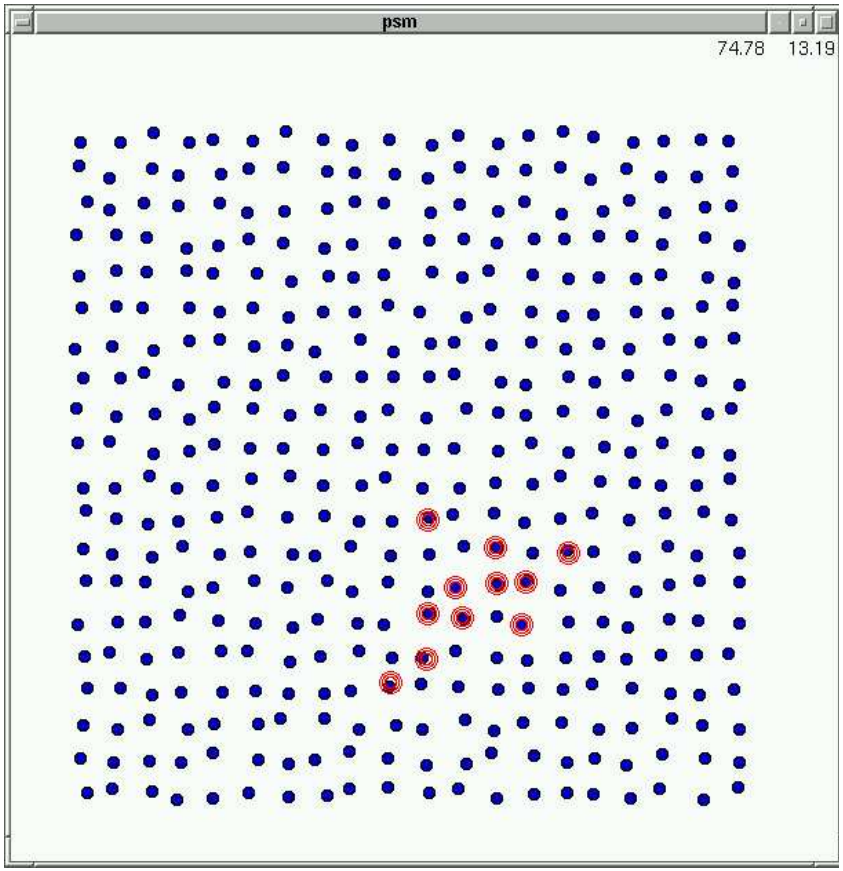


P

S

Search for the largest common pattern

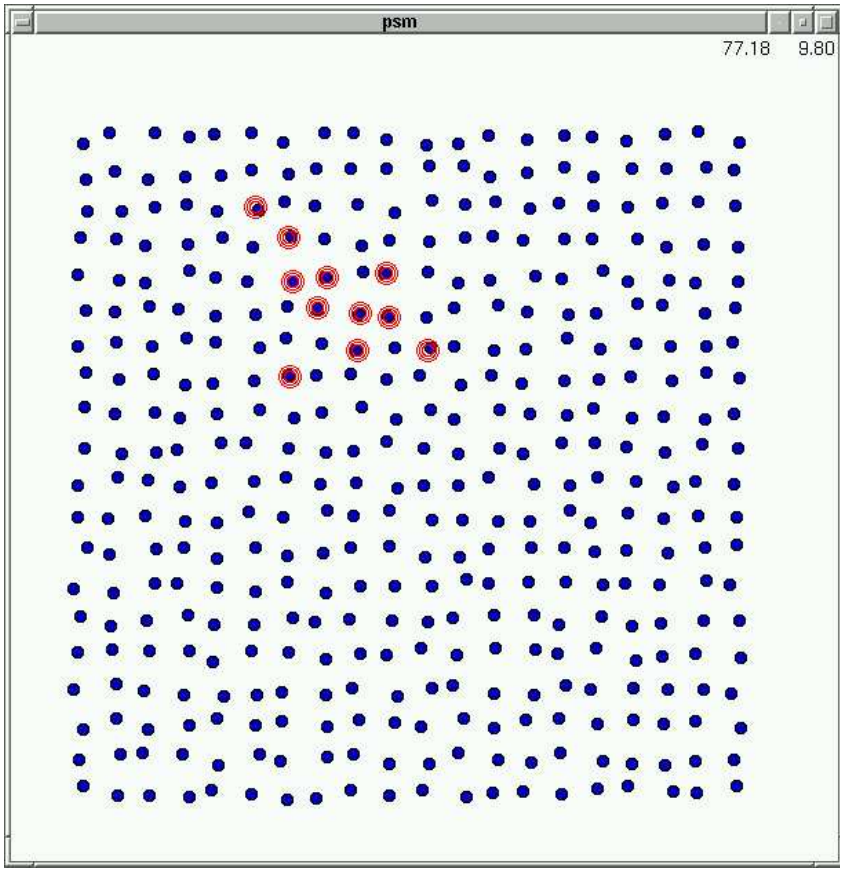


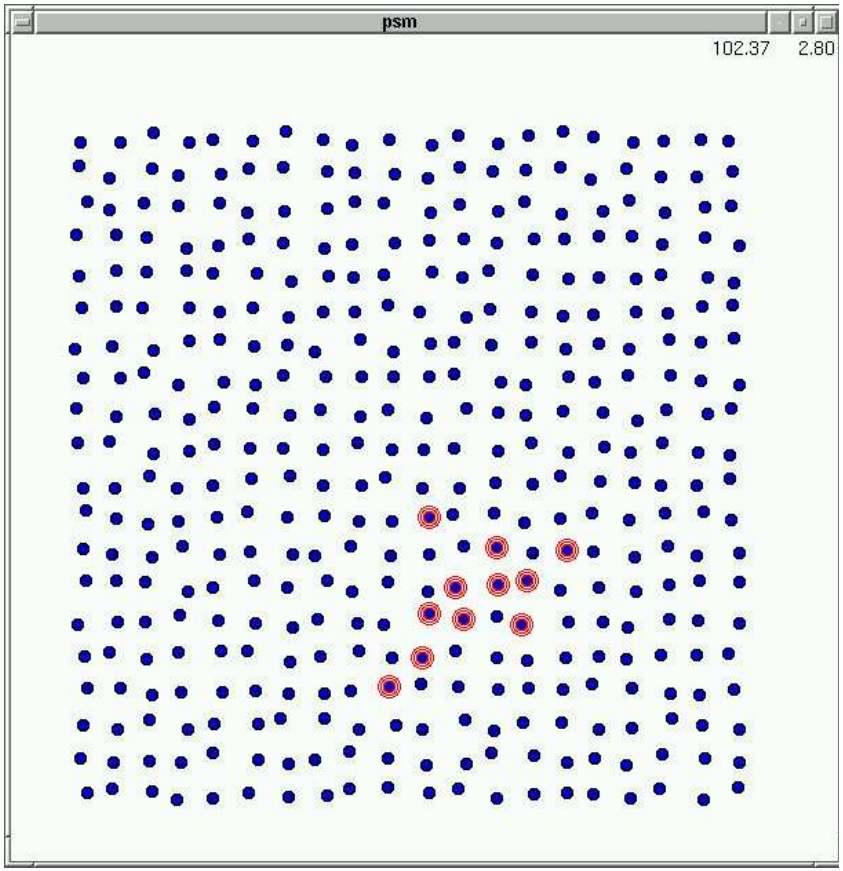


p/P

p/S

Search for the largest common pattern





s/P

s/S

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

Software

newer version

- almost 11.000 specific lines of C++
- uses the libraries:
  - mtl (*matrix template library*)
  - Gtkmm (graphical user interface)
  - own libraries

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