Advanced Algorithmics
Projects
Deadline: May 21st

Delay: 1 day = 20% of max points

Prerequisite for exam

2009 Spring

Expectations:

- Study the problem
- Implement
- Evaluate, Compare, Measure, …
- **Report – 2-3 p text; data; summaries; illustrations; tables; …**
- Your task is to make the project interesting to others: ***right questions; cool applications; novel ideas; desire to read; materials to complement next year courses. Find a clear objective and focus, state it!***
- **20-40h**
- **One person projects!**

Tasks

- Here is a list of some proposals
- You can propose your own.
- Or select some on your own
  - from international competitions
    - e.g. IOI (ACM) olympics finals series
    - implementation challenges from DIMACS, etc.
    - etc.

Sorting 1, 2, …

- Compare 2-3 linear time sorting methods (and parameters) with Quicksort from standard libraries

- Implement a Merge-Sort with in-place merging in O(n). Compare to some trivial implementations.

Tree layout

- Is in general quite well solved problem
- task is to implement some algorithms and visualise (ideally, using SWOG).

- Advanced: Add DAG properties
  - Example: Gene Ontology

Graph layout

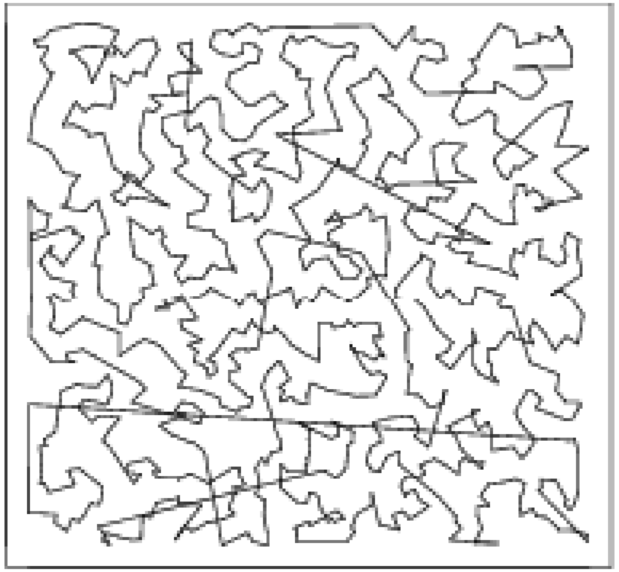
- Graph layout
  - “Physical Spring model” with some extra added constraints or specialised nodes for stars, cliques, connection strength, etc.

- Create a nr of criteria and try to minimize nr of crossings, area of graph, etc.

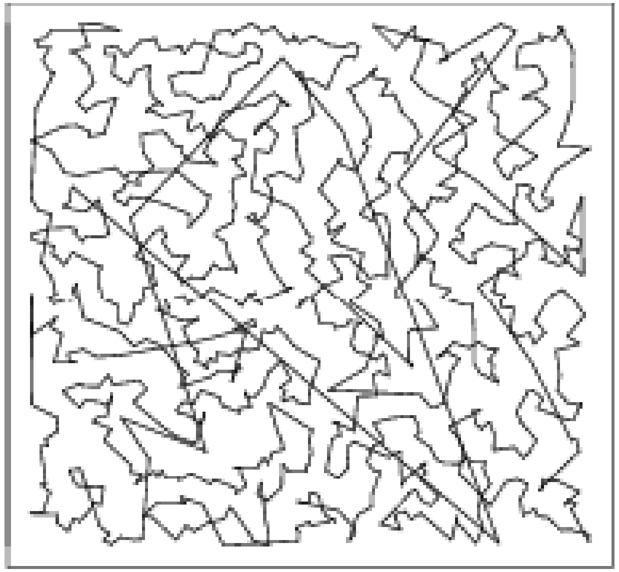
Public competitions

- Any of the public (past) competitions on optimisation, path-finding, sorting etc challenges

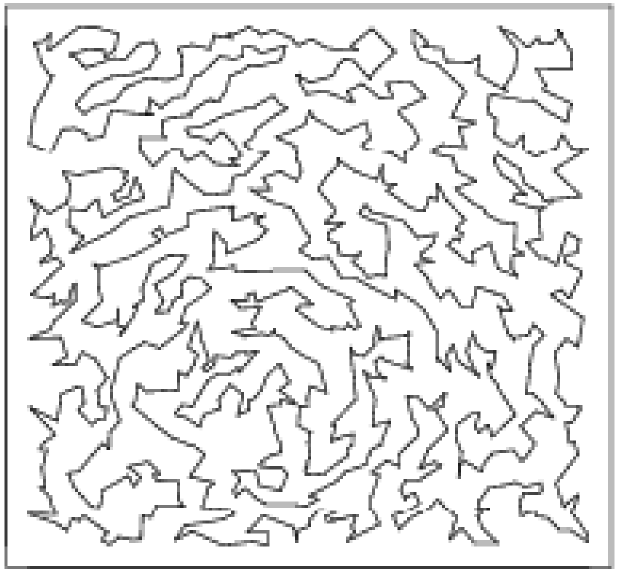
- Compare your results with best methods



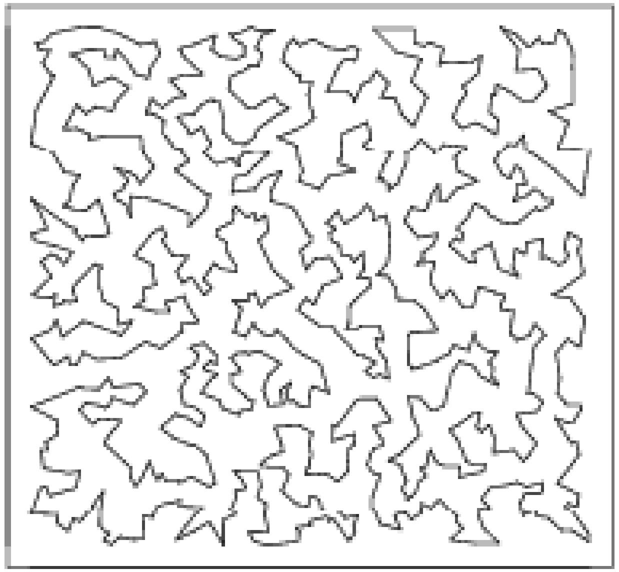
Greedy Tour



Nearest Neighbor Tour

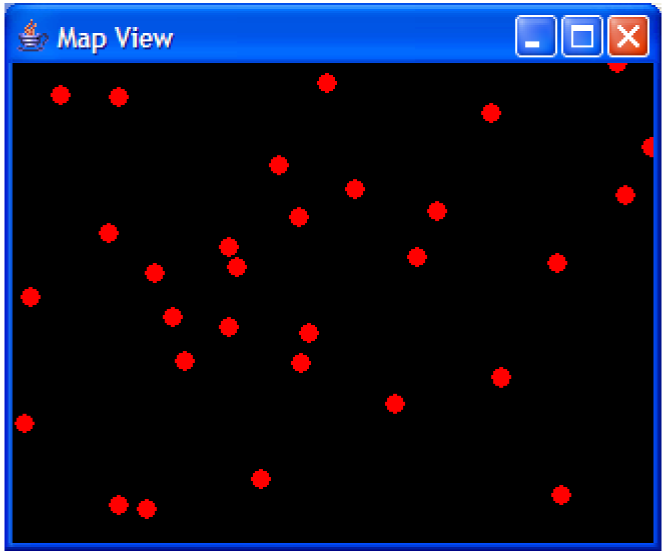


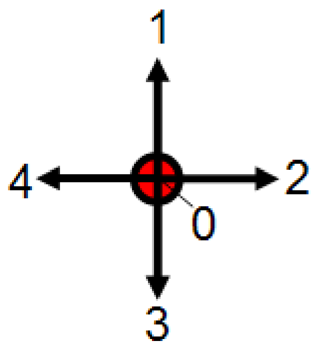
Savings Tour



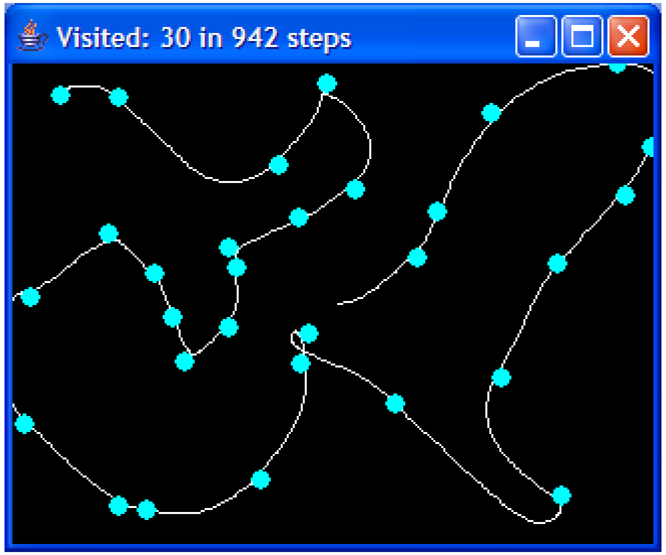
Optimal Tour

Visit all cities... - physically!



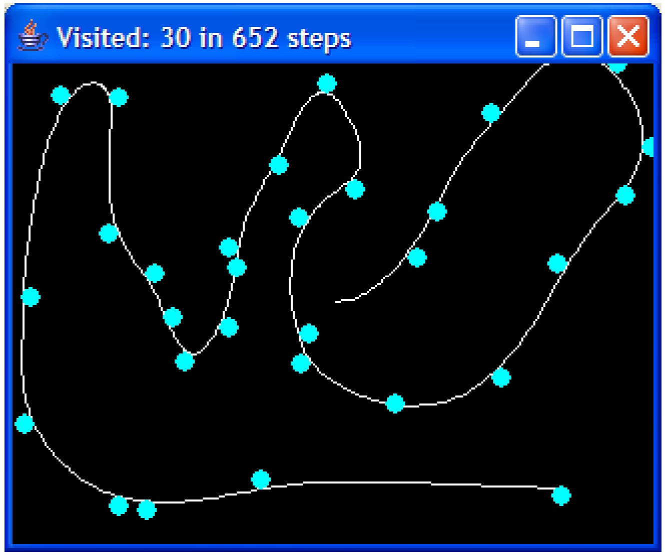


http://cswww.essex.ac.uk/staff/sml/gecco/PTSPComp.html
http://algoval.essex.ac.uk/ptsp/ptsp.html



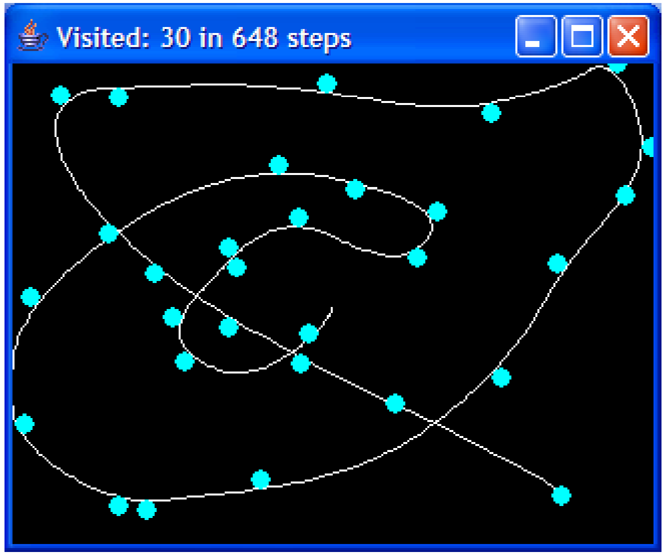
942, 941

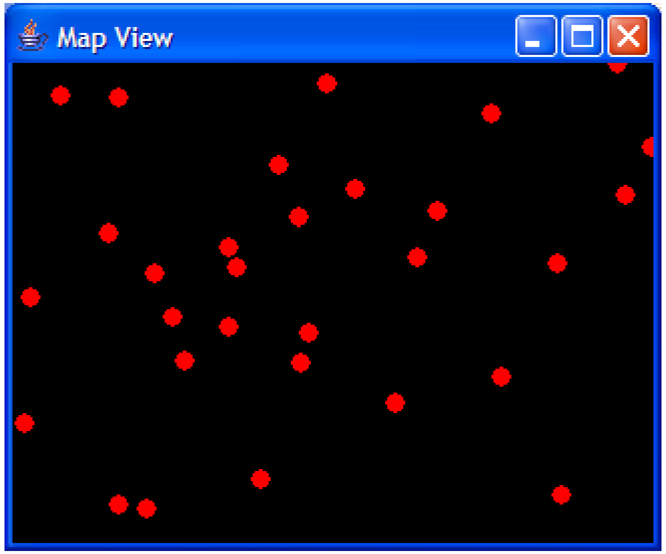
- 652, 652

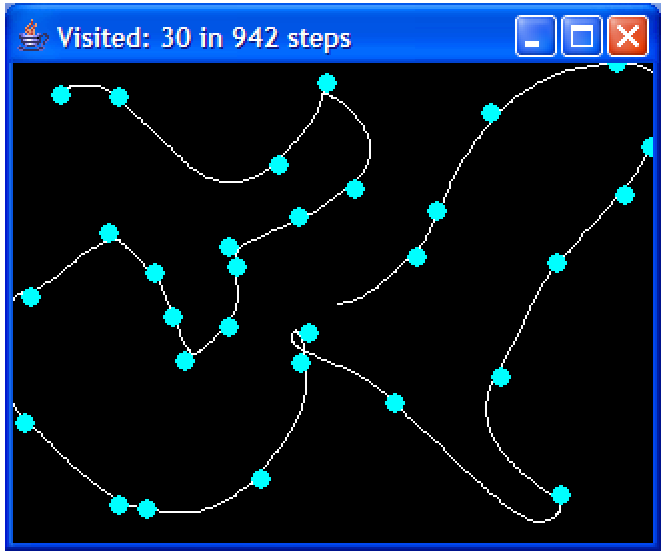


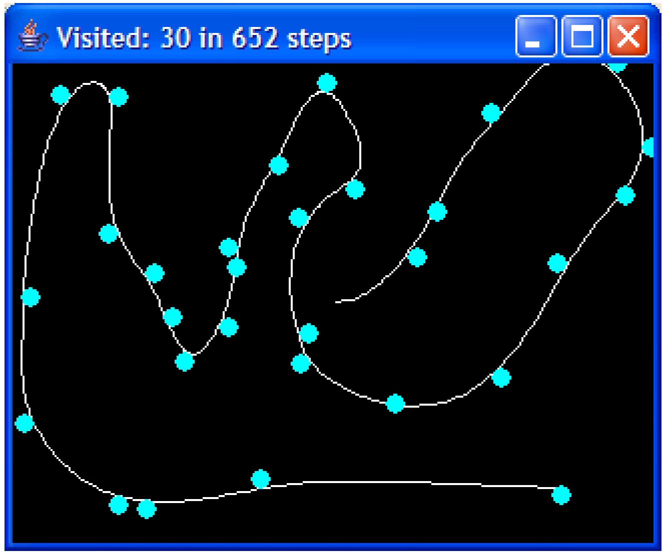
- 648, 636

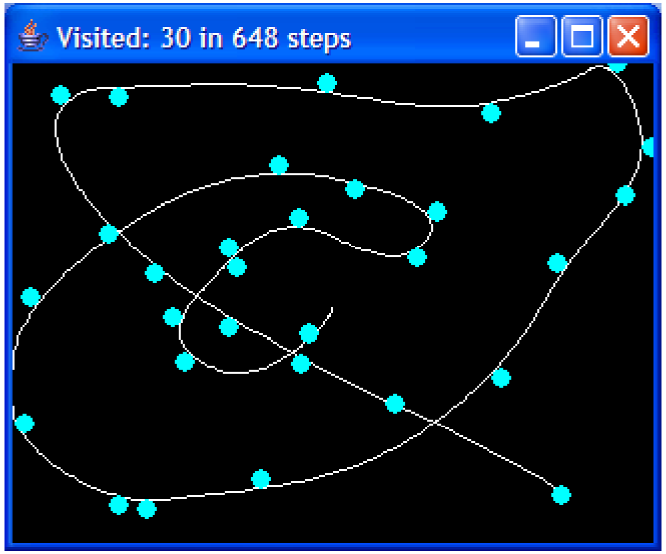
  3
  3
  3
  3
  3
  4
  3
  3
  4
  4
  4
  4
  3
  4
  4
  3
  3
  4
  4




Map View


Visited: 30 in 942 steps


Visited: 30 in 652 steps


Visited: 30 in 648 steps

Probabilistic Automated Bidding in Multiple Auctions

- A Linear best plan computation
- The following algorithm computes the best set of auctions in which to bid, given a bidding price r. It assumes that the time that it takes to get a quote or place a bid (written a) is the same across all auctions. Auctions are represented as integers.
- Algorithm in Appendix A.
  - http://eprints.qut.edu.au/1242/1/ecr.pdf

Compound word generator

- Given a set of words/strings
- Generate maximally overlapping artificial compound words.
  - kahvanäguripäevapiltnik
- **Minimal length**, **maximum nr of words** concatenated with overlaps. (see bonus task from exercises)
- Extra bonus 1: use natural word boundaries
- Extra bonus 2: **palindromic**! (2-way search)

Sudoku solver

- Sudoku solver – and solving strategies
  - constraint satisfaction
  - backtracking
  - depth-first; best-first; …
  - http://en.wikipedia.org/wiki/Algorithmics_of_sudoku

15-puzzle

- Solver for 15-puzzle
  - goal is to find shortest path solutions
  - try heuristics
  - different sizes of the problem

- *Comment: In case of a well-studied problem, expectations on project achievements, comparisons, reporting, a good tutorial style, etc are much higher; it's harder to show your own creativity.*

2-D shortest path finder

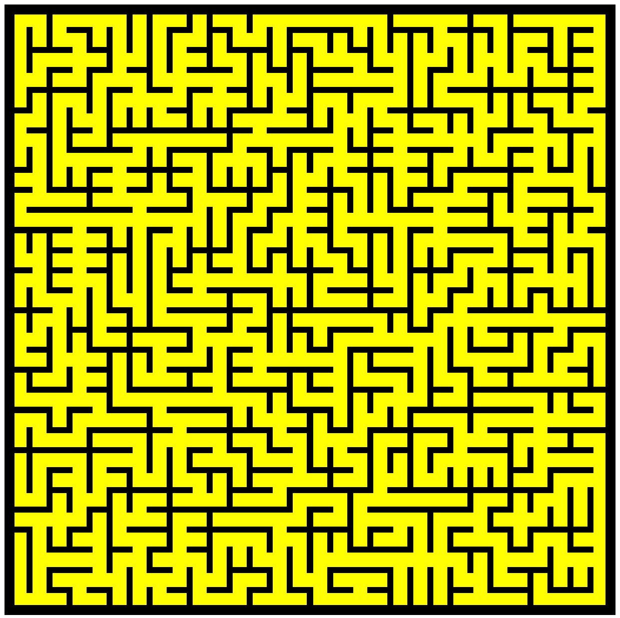
- graph with physical 2-D coordinates
  – any GPS navigator, Map server, etc.
  – find or create maps

- Shortest path finding using A* and/or other algorithms

- Weighted, directed graph

Maze solver

– try to minimize the length of traversed corridors
– Visualise different strategies
– e.g. use the random labyrinths from course,
    - http://emu.at.mt.ut.ee/u/vilo/Algorithmics/maze.cgi?seed=&N=10&px=10&swog=ON&.submit=Submit+Query&.cgifields=swog



I SPEXS: general algorithm

1. S = input sequences (||S||=n)
2. e = empty pattern, e.pos = {1,...,n}
3. enqueue(order , e)

4. **while p** = dequeue(order)
5. **generate all allowed extensions p' of p (& p'.pos)**
6. enqueue(order, p', **priority**(p'))
7. enqueue(output, p', **fitness**(p'))

8. **while** p = dequeue(output)
9. Output p

Jaak Vilo: Discovering Frequent Patterns from Strings.
Technical Report C-1998-9 (pp. 20) May **1998**. Department of Computer Science, University of Helsinki.

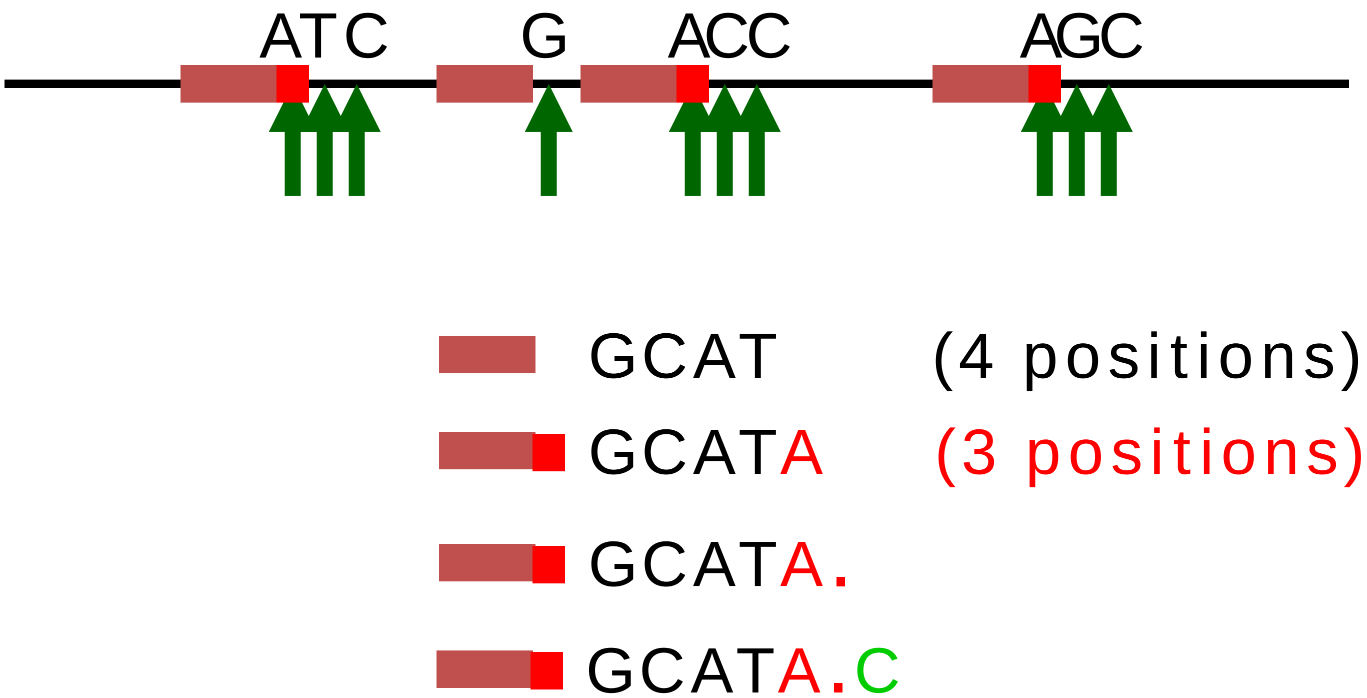
Jaak Vilo: Pattern Discovery from Biosequences
PhD Thesis, Department of Computer Science, University of Helsinki, Finland.
Report A-2002-3 Helsinki, November 2002, 149 pages

Applications in bioinformatics:

-Gene regulation (1998: 255+ citations, 2000: 73 cit)

-Functional elements in proteins (2002: 32 cit)

Sequence patterns: the basis of the SPEXS



- priority: *e.g.* depth-first; breadth-first; best-first; A*, beam search, …

- fitness: most frequent; max(freq * length); …

Similarity join

- From two sets of objects
  - find the best matching pairs
- E.g. two photos from slightly changed angle
  - identify “same” points on two images
  - can have applications in 3-D stereo imaging; photo stiching (panoramic photos), etc.

Graph edit distance

- Calculate edit distance between graphs:
  - add/delete nodes, edges
  - match nodes (e.g. by partially matching labels)

  - Q: How similar are two graphs

Graph: SESE

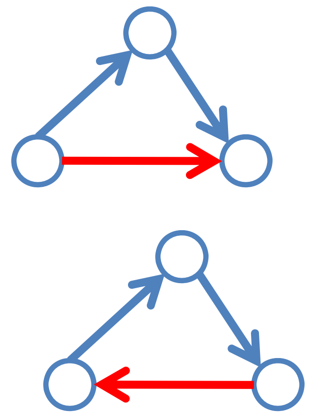
- Identify region(s) of **Single Entry, Single Exit**.

- I.e. find subgraphs that can be "collapsed" and studied further recursively, for example

- Marlon: Here is a set of test cases you could give to the students who choose the project on "identifying SESE regions". The rar file attached contains a README file that explains the contents of the folder and the format used to encode the process graphs.

Graph: counting features

- Count a nr of times certain features exist in graphs; find “most frequent features”
- E.g. small feedback-triangles; feed-forward triangles, etc.
- Hint: count nr of smallest features; add certain feature and recalculate their frequencies.
- Keep expanding the most frequent features (similar to SPEXS idea)



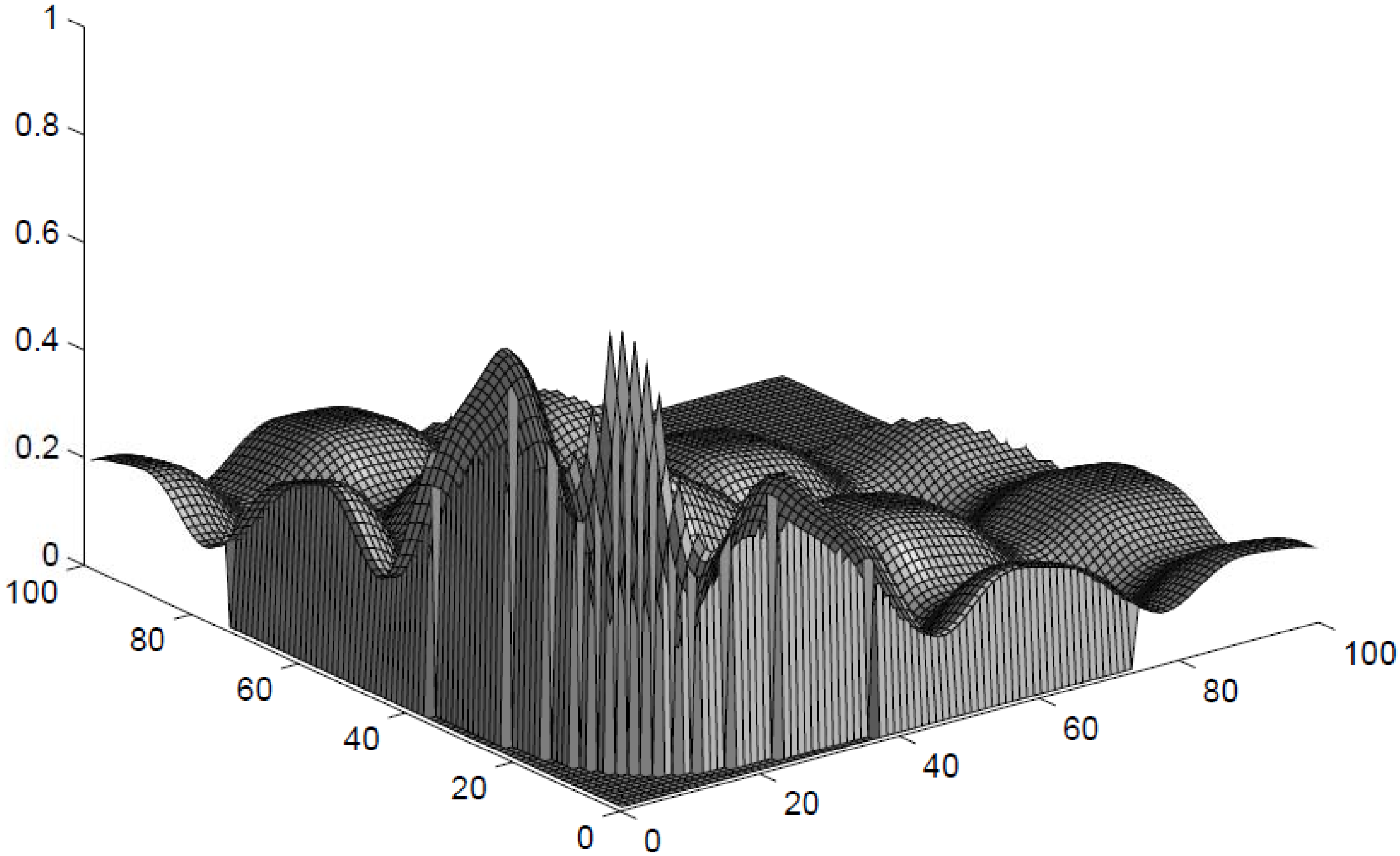
Graph

- Implement Page Rank algorithm
  - Evaluate problem sizes, provide examples

  - Abstract graphs
  - Method could be used on scientific literature, social networks, etc…
  - Can you add more features, prevent "spammers"?

Automaton -> Regular expression

- Implement a simple automaton->RE mapper
- Attempt to find ways to create shortest RE-s
  - case study: RE->NFA->DFA->Minimize->RE
  - **can you still get back the original RE ?**
- Study problem, identify potential ideas…



The graph of function $G2$ for $n = 2$. Infeasible solutions were as

Grammatical Inference

- Bonus task from this week: find smallest automaton consistent with all positive examples, and none of the negatives…

- Exhaustive search (?)
- Propose some search strategy (local search; simulated annealing; GA)
- Focus on idea (partly on examples on paper)

TSP using local search and Simulated Annealing