Evolutionary Computation

2023/24

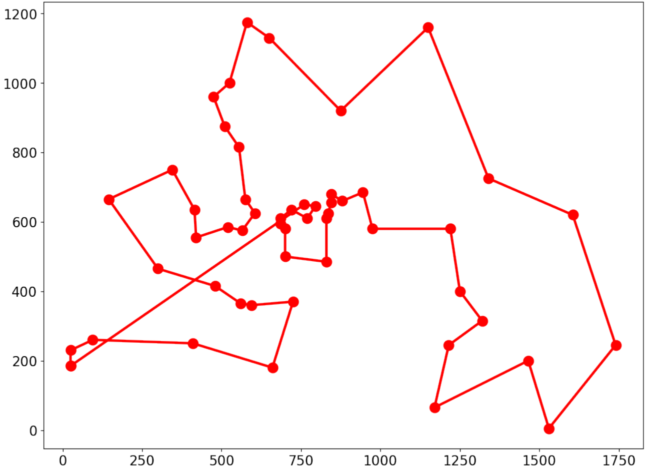
Master Artificial Intelligence

Arno Formella

Departamento de Informática
Escola Superior de Enxeñaría Informática
Universidade de Vigo

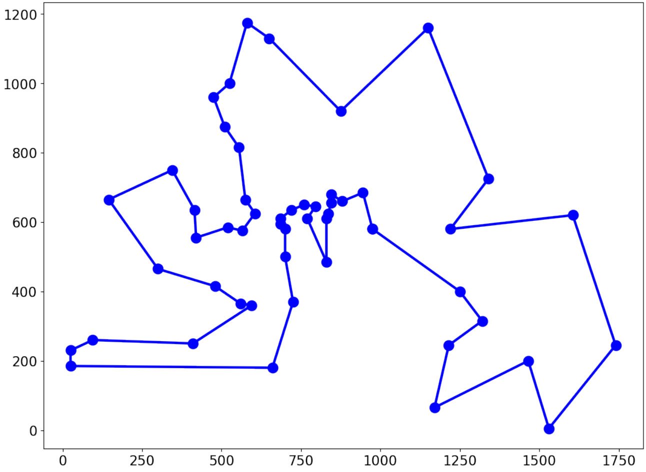
23/24

traveling salesperson problem: first impressions



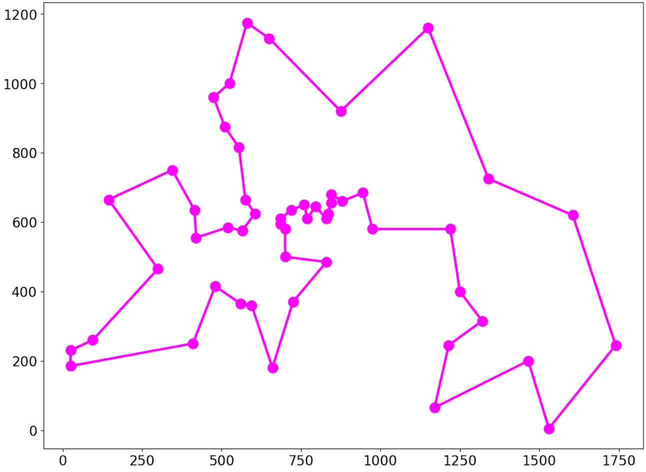
a closest-neighbor tour: 8.49% relative error

traveling salesperson problem: first impressions



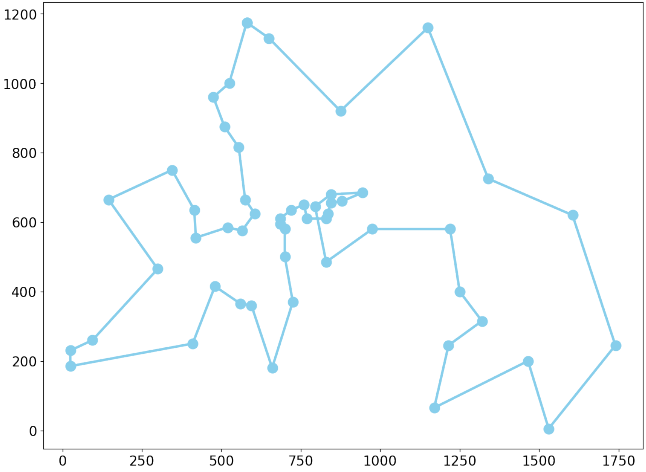
a pair-center tour: 7.28% relative error

traveling salesperson problem: first impressions



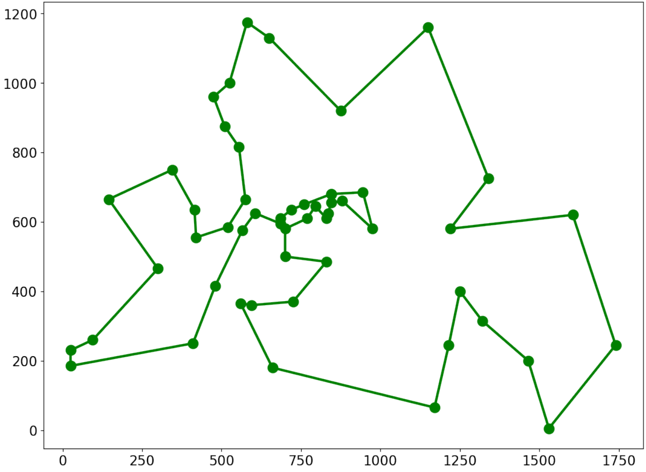
the best tour (known for this example): 0% relative error

traveling salesperson problem: first impressions



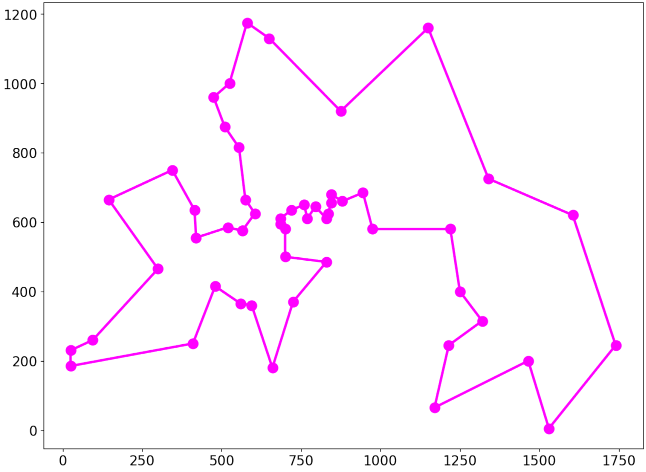
a quick tour (with Monte Carlo): 2.32% relative error

traveling salesperson problem: first impressions



a genetic algorithm tour: 8.37% relative error

traveling salesperson problem: first impressions



the best tour (known for this example): 0% relative error

lower bounds on tour length of TSP

- simple bound: sum of minimal distances to neighbors
- Held-Karp bound
  (typically comes close to 1% on random instances and below 2% on TSPLIB, arguments for the bound are quite complicated)

upper bounds on tour length of TSP

- tour around minimal spanning tree yields $\leq 2 \cdot L_{opt}$
  runtime $\mathcal{O}(n^2)$
- Christofides algorithm yields $\leq 1.5 \cdot L_{opt}$
  runtime $\mathcal{O}(n^2 \log n)$

known solving algorithms for TSP

- brute force exhaustive search $\mathcal{O}(n!)$
  (quite easy to implement)
- Bellman-Held-Karp dynamic programming for Euclidean TSP
  $\mathcal{O}(n^2 2^n)$ time and $\mathcal{O}(n2^n)$ space
  (not covered here, please refer to advanced algorithms in computer science)
- state-of-the-art solver Concorde.
- state-of-the-art approximative solver LKH.

known approximation heuristics for TSP

start with some small tour generated by a simple heuristic, then:

- use 2-opt moves (modifing tour by changing two edges, for instance, to eliminate crossings in Euclidean TSP);
- or use 3-opt moves (modifing tour by changing three edges);
- or use Lin-Kernighan heuristic algorithm
  (variable mixture of 2-opt and 3-opt moves),
  currently the best known heuristic strategy.

TSP on a grid

let's take a look at a regular $m \times n$ grid (e.g. checker board)

- an optimal tour on a regular grid is easy to build
- optimal length:
  - $L_{opt} = n \cdot m$ if n or m even
  - $L_{opt} = n \cdot m - 1 + \sqrt{2}$ if $n \cdot m$ odd
- there are many! optimal tours

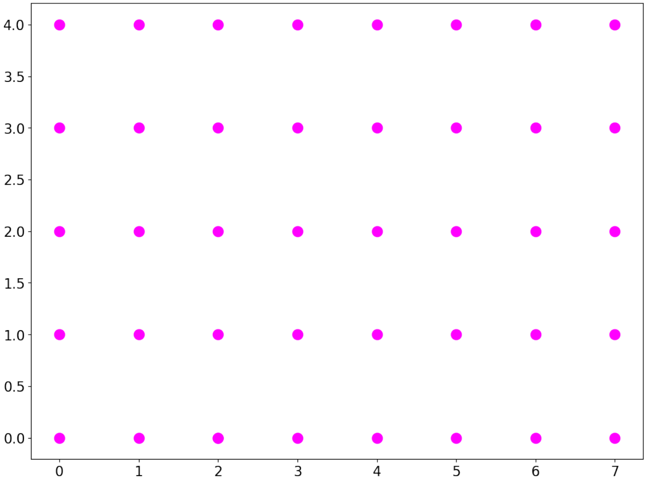
More on traveling salesperson problem

- The version of the TSP problem as shown until now is a special case of a more general problem definition:
- Let $G = (V, E)$ be a graph. V are the nodes (or locations), E are the edges (or connections) with some weight (e.g., distance, time, cost).
- Goal: find a minimal tour through all nodes.
- Particularly we talked about the Euclidean TSP, where the nodes are points in the Euclidean plane and the distance among all pairs, hence complete graph, is just the Euclidean distance.
- One step to be more general is, just require the triangular condition to be met (then, possibly, the pair-center approach cannot be used as we have no distances for the centers), this is called the metric TSP (mTSP).
- Moreover, the distances might be asymmetric, i.e., going in one direction is different from going in the other (ATSP).
- or we have additional conditions: open loop, arrival time windows, asymmetric distances, interrupted tours, etc.

More recent results on the eTSP

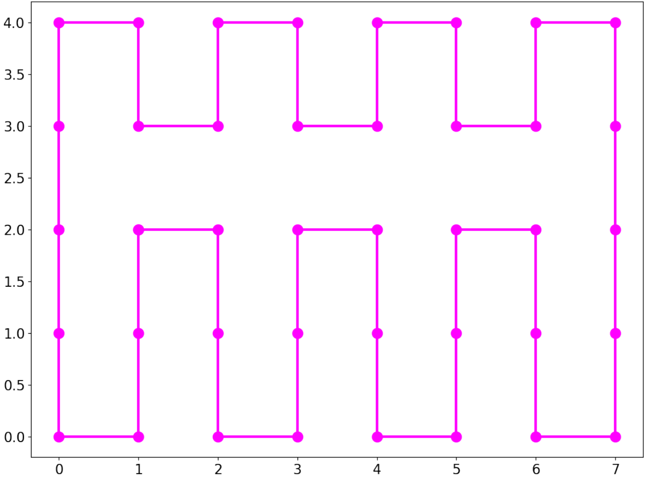
- In the 90's it was shown that eTSP can be solved in $\mathcal{O}(2^{O(\sqrt{n}\log n)})$.
- In the 10's this was improved to $\mathcal{O}(2^{\sqrt{n}})$, and with certain arguments that further improvement may be very unlikely.
- One recent result of complexity theory is that eTSP has a polynomial time approximation scheme (PTAS) of $\mathcal{O}_{\varepsilon,d}(n\log n)$ (with fixed error ε and fixed dimension d), however, an implementation is not available (to my knowledge).
- This has been improved to $\mathcal{O}_{\varepsilon,d}(n)$ with high probability (2013).

traveling salesperson problem: first impressions



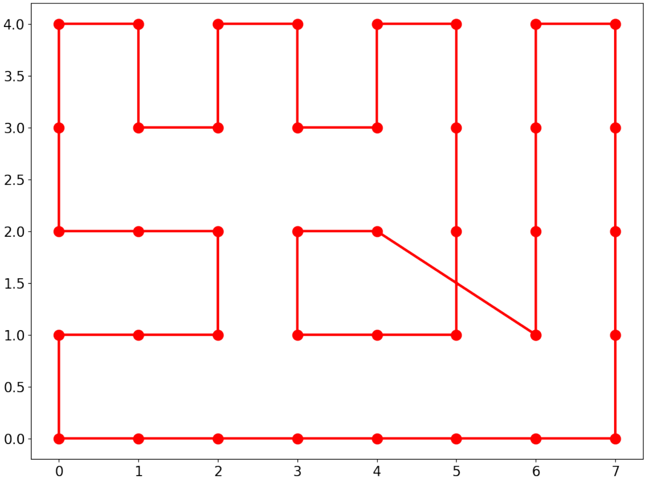
the cities distributed geographically

traveling salesperson problem: first impressions



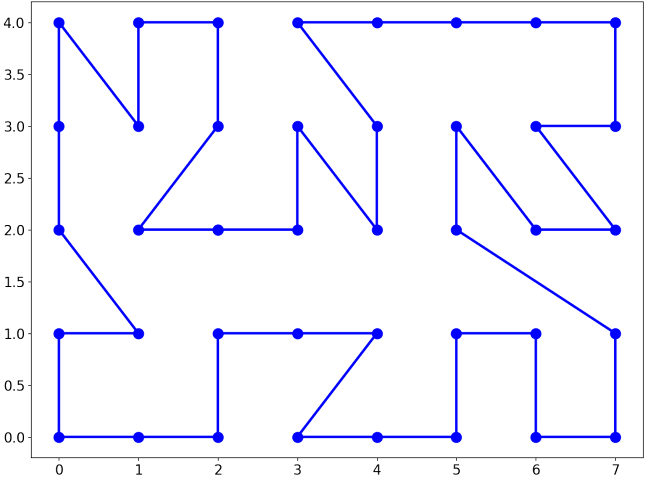
a best tour (trivial for this example): 0% relative error

traveling salesperson problem: first impressions



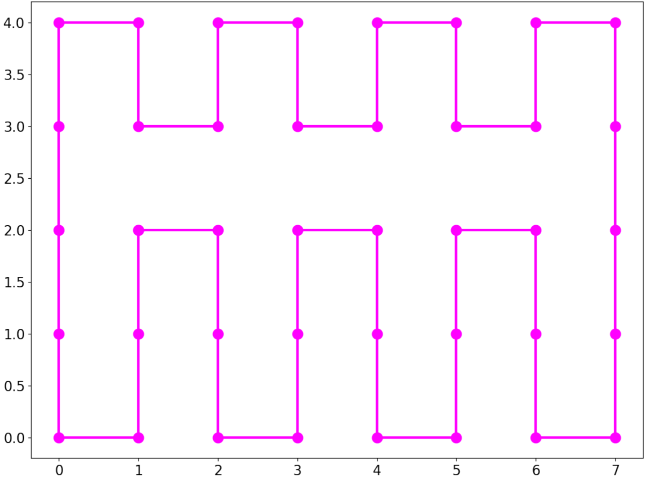
a closest-neighbor tour (with Monte Carlo): 3.09% relative error

traveling salesperson problem: first impressions



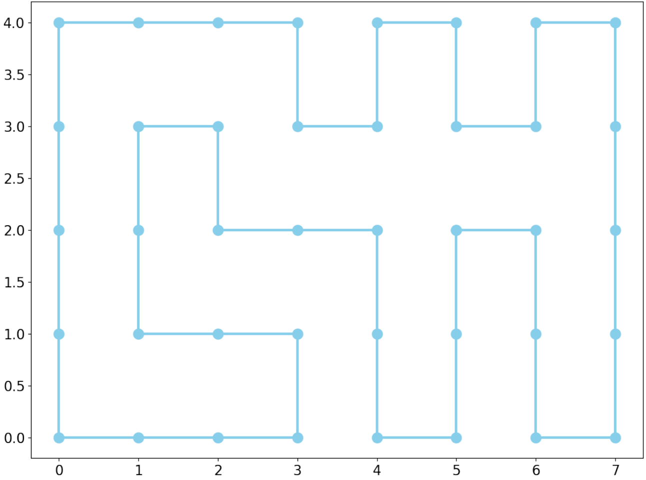
a pair-center tour: 14.46% relative error

traveling salesperson problem: first impressions



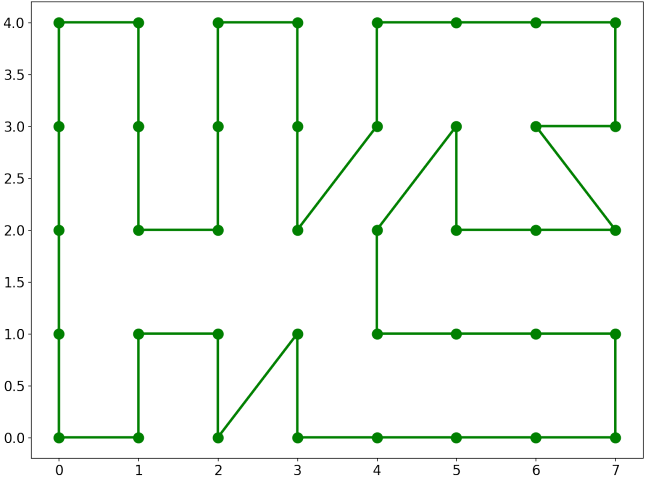
a best tour (trivial for this example): 0% relative error

traveling salesperson problem: first impressions



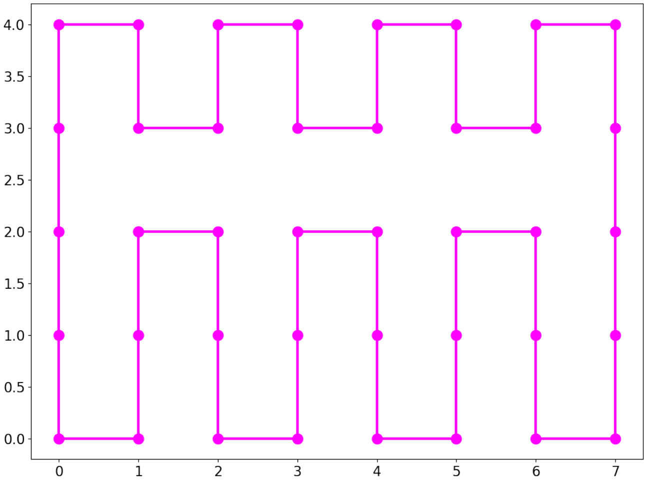
a quick tour (with Monte Carlo): 0.00% relative error

traveling salesperson problem: first impressions



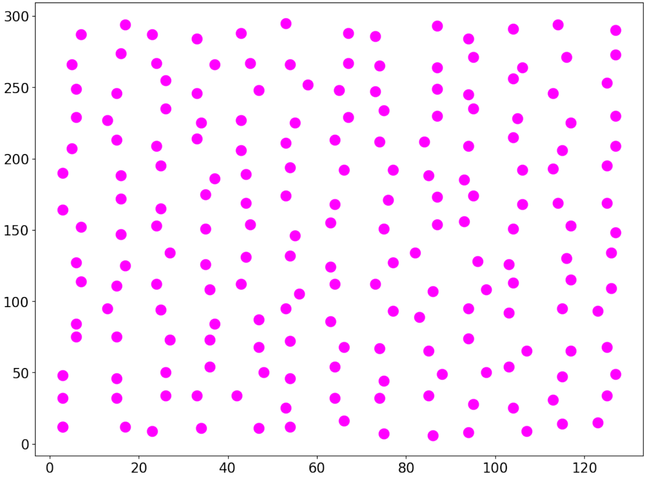
a genetic algorithm tour: 4.14% relative error

traveling salesperson problem: first impressions



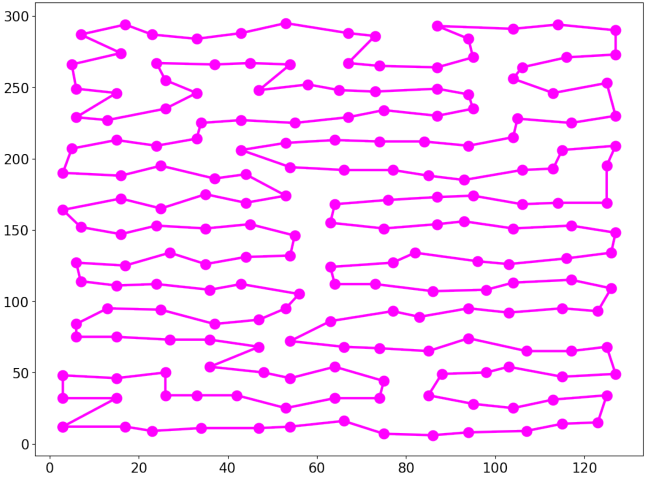
a best tour (trivial for this example): 0% relative error

traveling salesperson problem: first impressions



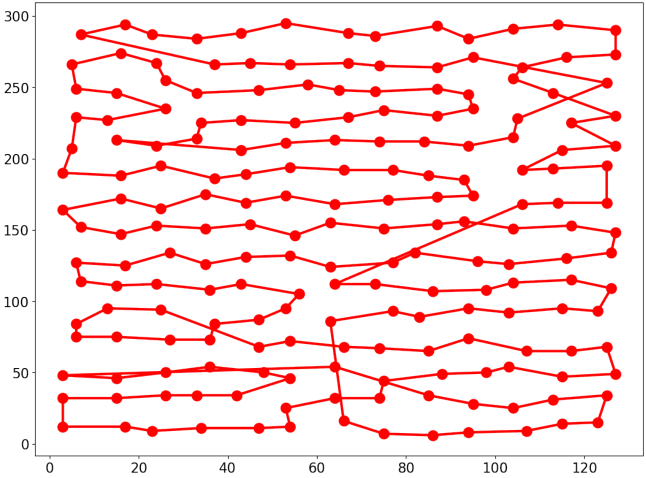
the locations distributed in the plane

traveling salesperson problem: first impressions



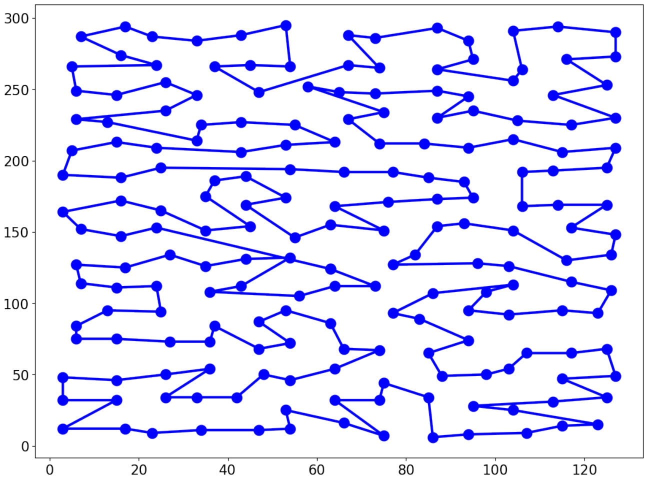
the best tour (known for this example): 0% relative error

traveling salesperson problem: first impressions



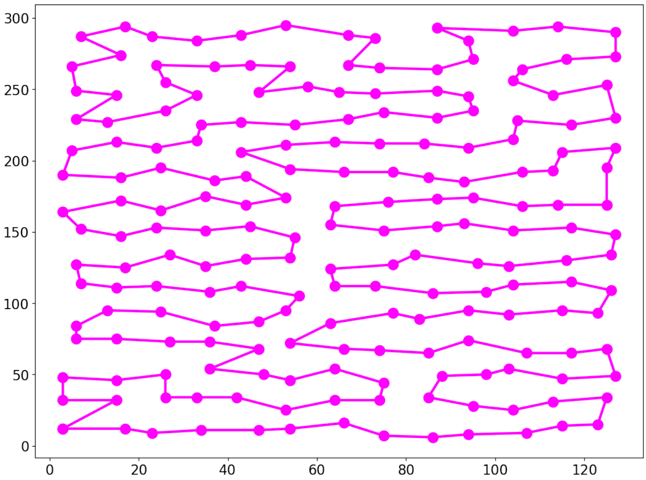
a closest-neighbor tour (with Monte Carlo): 10.23% relative error

traveling salesperson problem: first impressions



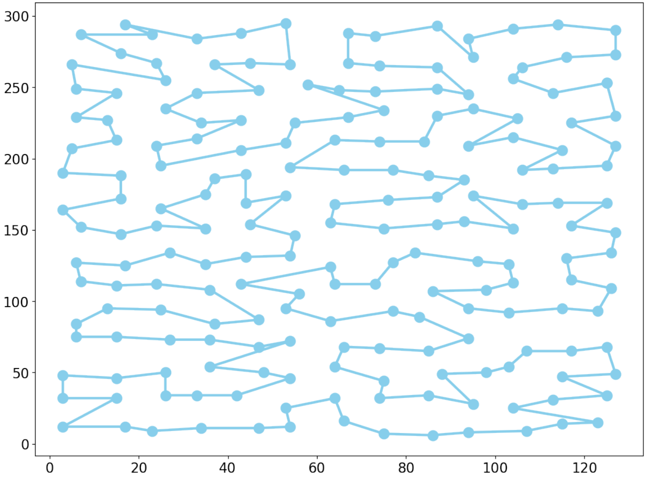
a pair-center tour: 13.93% relative error

traveling salesperson problem: first impressions



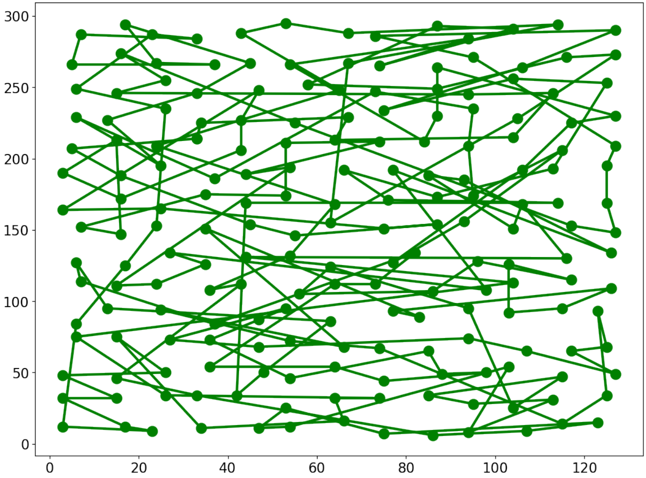
the best tour (known for this example): 0% relative error

traveling salesperson problem: first impressions



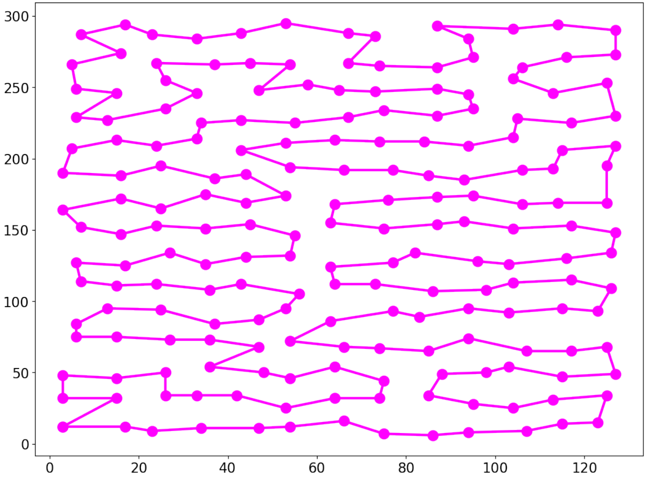
a quick tour (with Monte Carlo): 9.66% relative error

traveling salesperson problem: first impressions



a genetic algorithm (GA) tour: 205.65% relative error

traveling salesperson problem: first impressions



the best tour (known for this example): 0% relative error

errors for the above heuristics

problem	heuristic	relative error
berlin52	closest neighbor tour	8.49
	quick tour	2.32
	pair-center tour	7.28
	genetic algorithm tour	8.37
	pair-center tour improved	0.00
	Lin-Kernighan tour	0.00
rat195	closest neighbor tour	10.23
	quick tour	9.66
	pair-center tour	13.93
	genetic algorithm tour	205.65
	pair-center tour improved	1.16
	Lin-Kernighan tour	0.00
block40	closest neighbor tour	3.09
	quick tour	0.00
	pair-center tour	14.46
	genetic algorithm tour	4.14
	improved pair-center tour	0.00
	Lin-Kernighan tour	0.00

Your goal: make GA-tour consistently better than pair-center tour or quick tour.

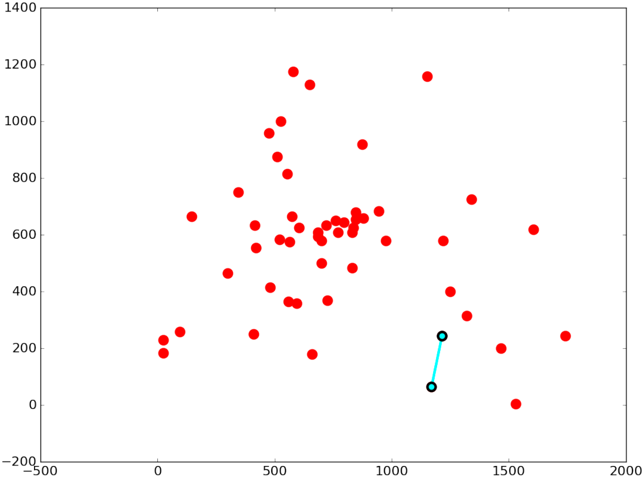
How does the closest-neighbor algorithms work?

The classical closest-neighbor algorithm is a greedy algorithm with a small random component:

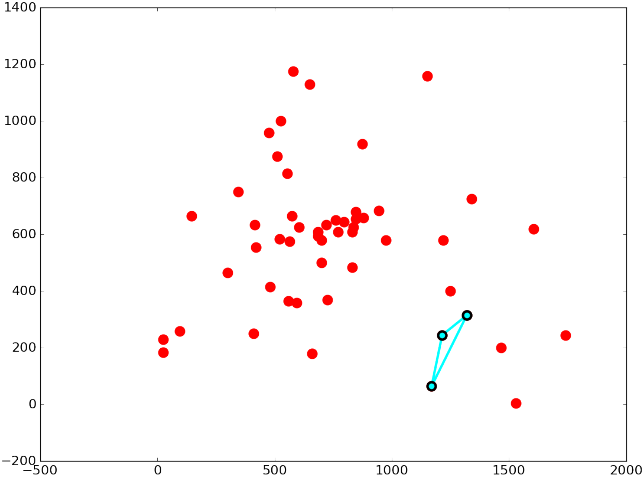
- select a random city
- while there are still unconnected cities
  - connect to the closest unconnected neighbor
  - use a random tie break
- connect the first with the last city

- runtime is $\mathcal{O}(n^2)$,
- can be run in Monte Carlo fashion keeping the shortest tour
- worst tour may have a length up to $0.5 \cdot \log n \cdot L_{opt}$

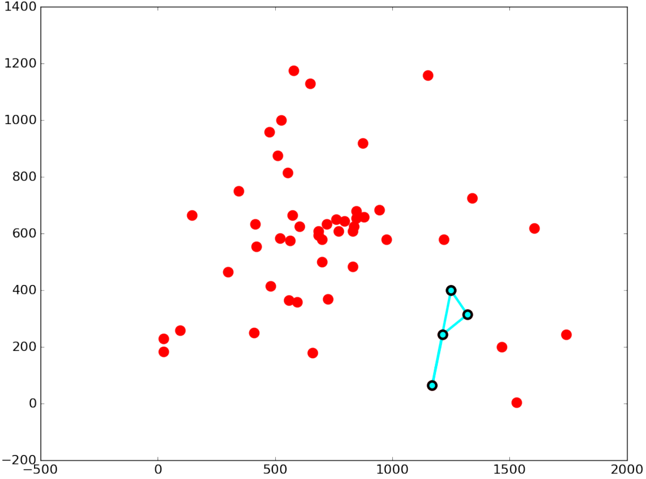
animation of the closest-neighbor algorithm



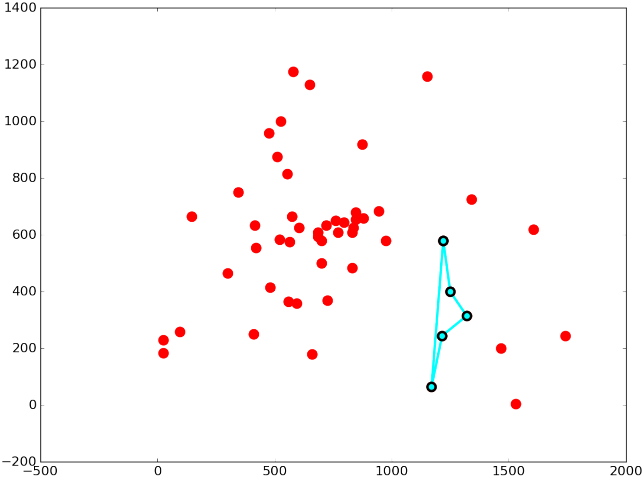
animation of the closest-neighbor algorithm



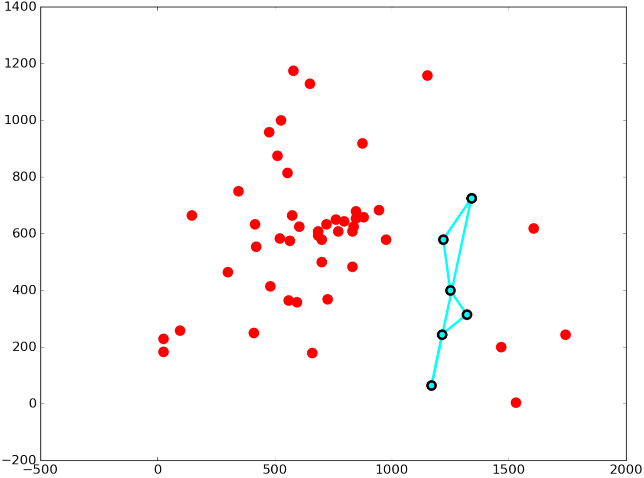
animation of the closest-neighbor algorithm



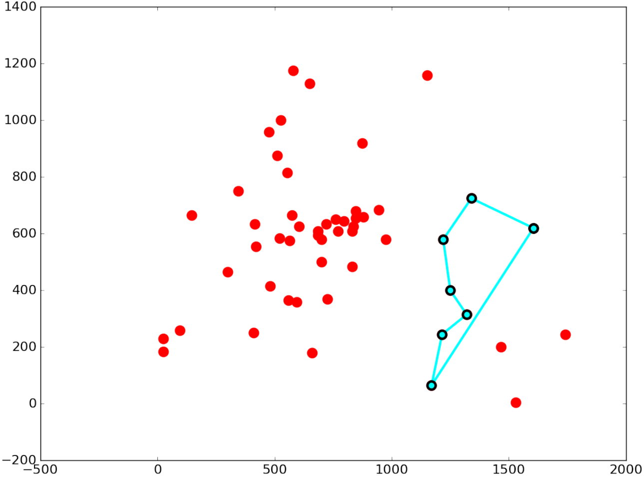
animation of the closest-neighbor algorithm



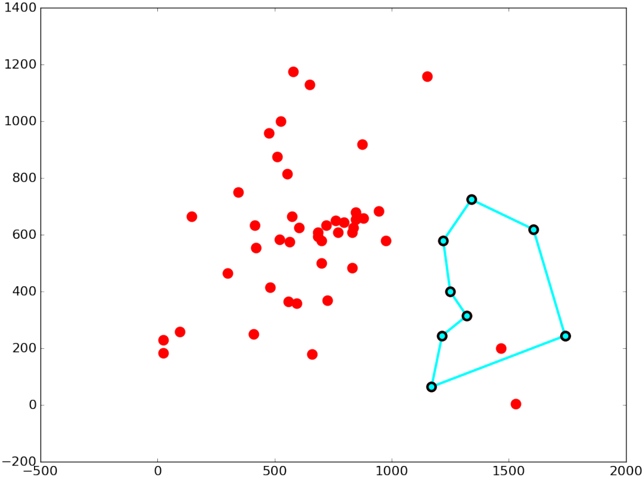
animation of the closest-neighbor algorithm



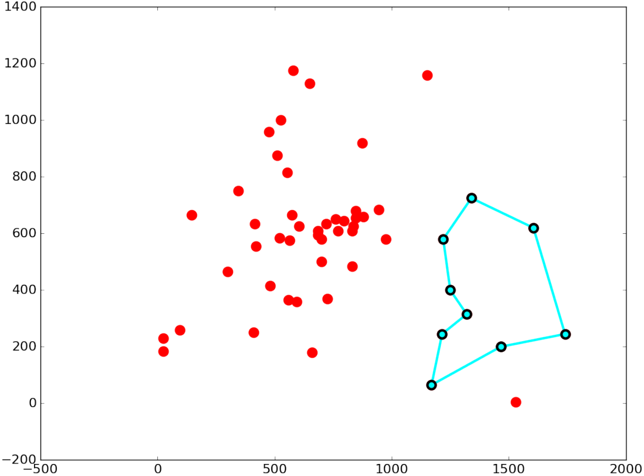
animation of the closest-neighbor algorithm



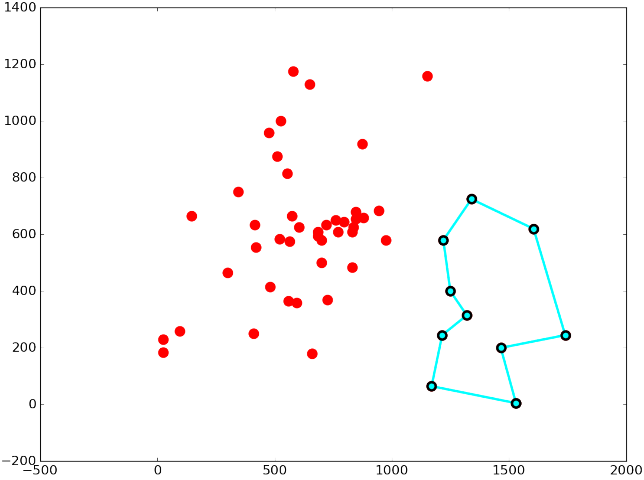
animation of the closest-neighbor algorithm



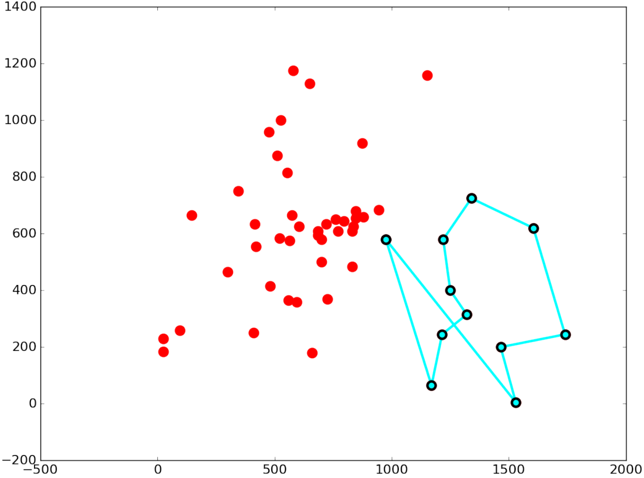
animation of the closest-neighbor algorithm



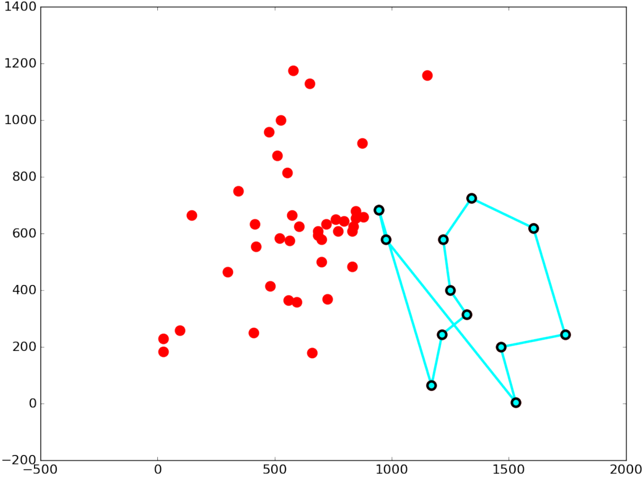
animation of the closest-neighbor algorithm



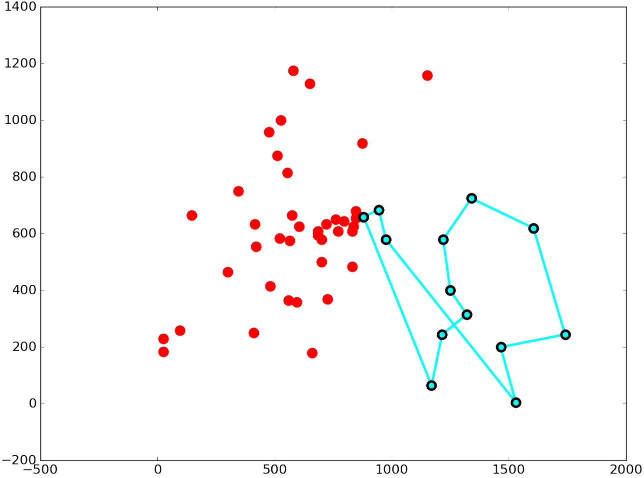
animation of the closest-neighbor algorithm



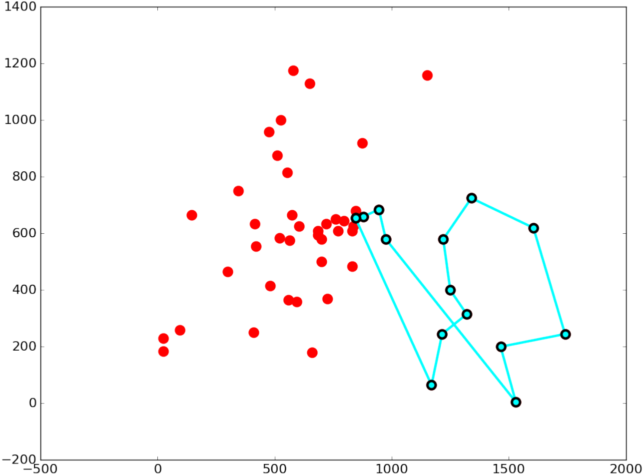
animation of the closest-neighbor algorithm



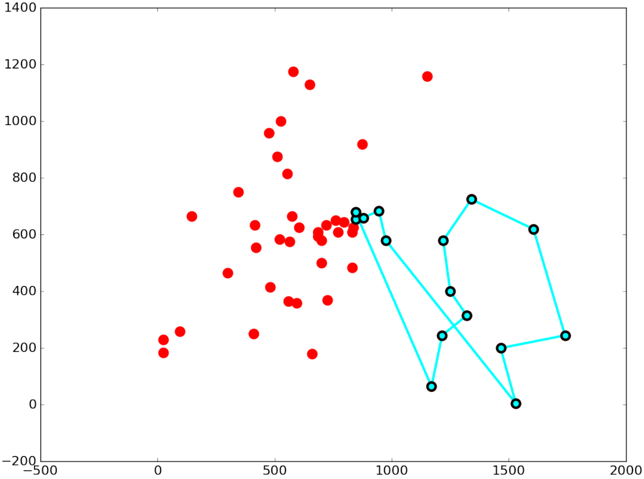
animation of the closest-neighbor algorithm



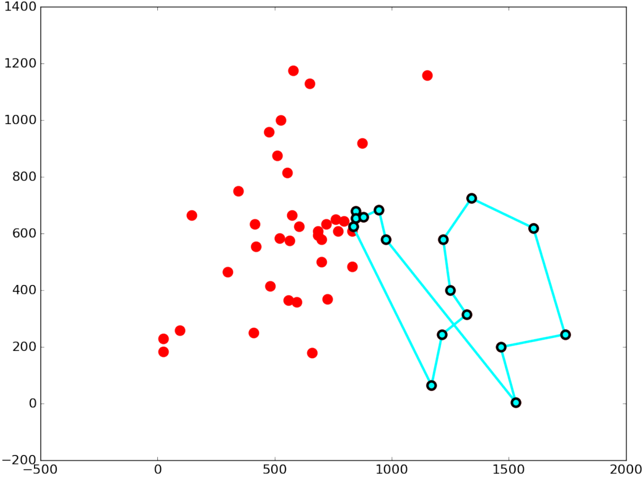
animation of the closest-neighbor algorithm



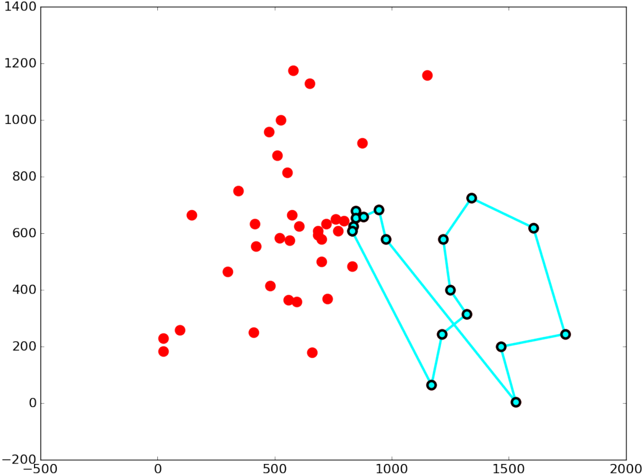
animation of the closest-neighbor algorithm



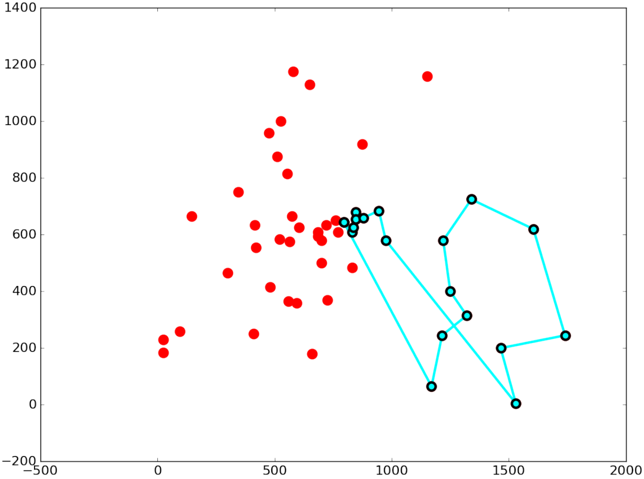
animation of the closest-neighbor algorithm



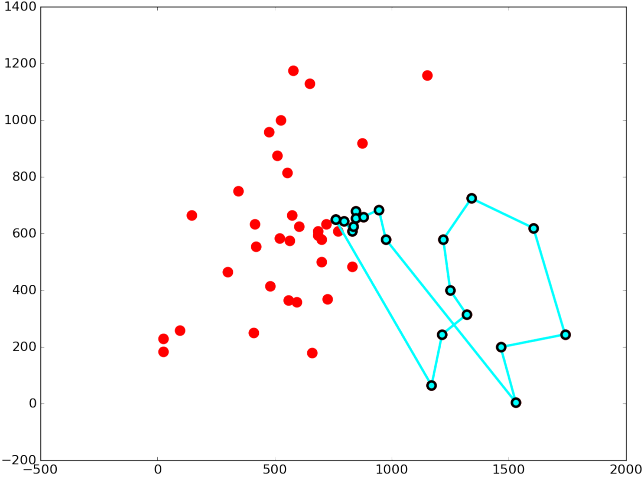
animation of the closest-neighbor algorithm



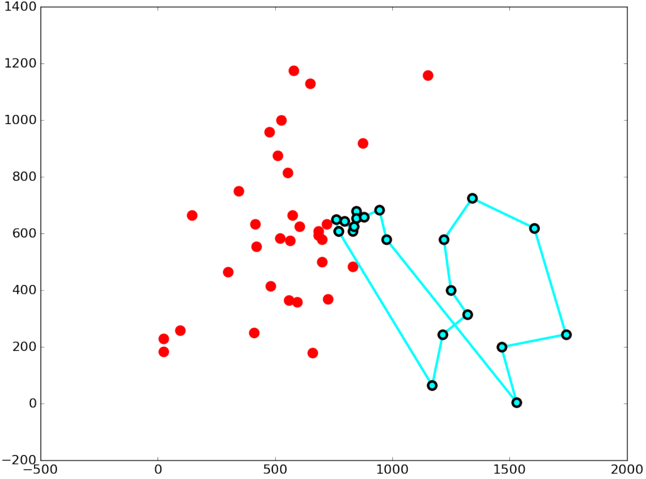
animation of the closest-neighbor algorithm



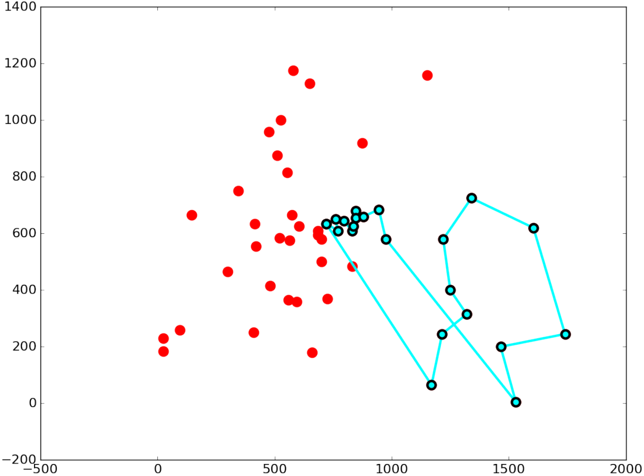
animation of the closest-neighbor algorithm



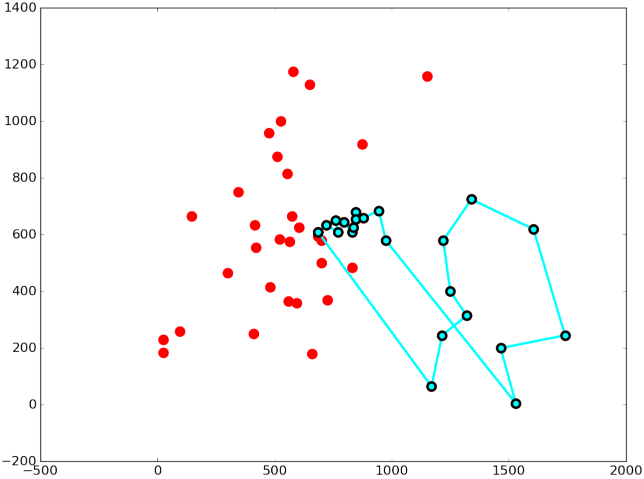
animation of the closest-neighbor algorithm



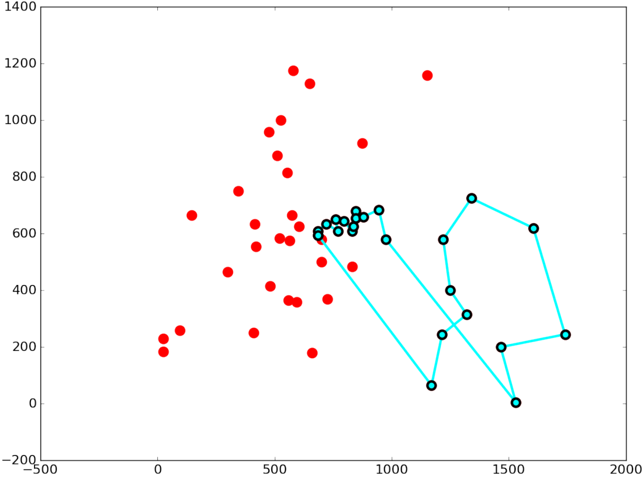
animation of the closest-neighbor algorithm



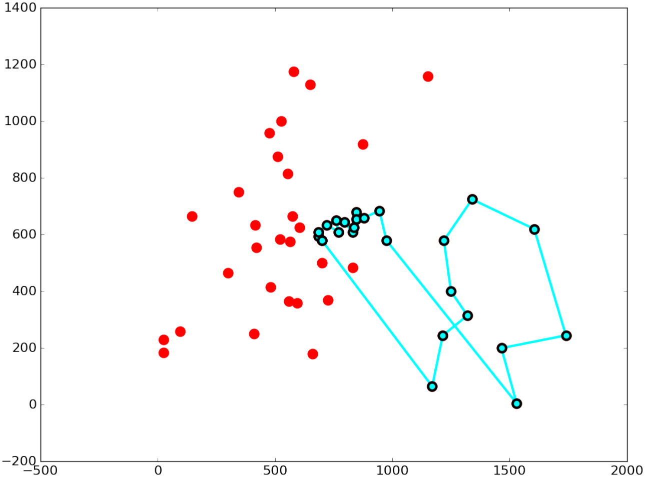
animation of the closest-neighbor algorithm



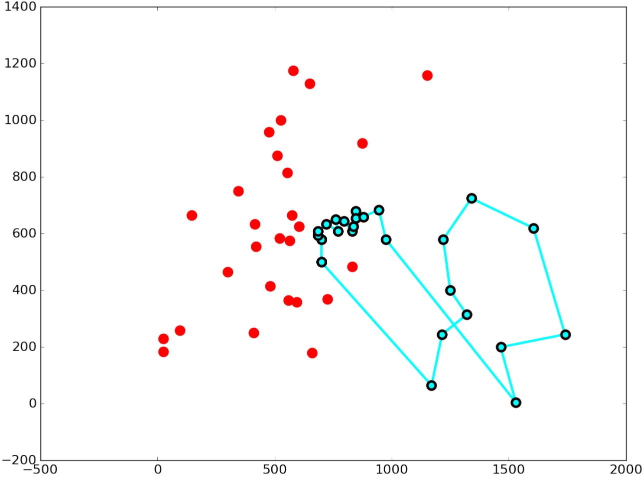
animation of the closest-neighbor algorithm



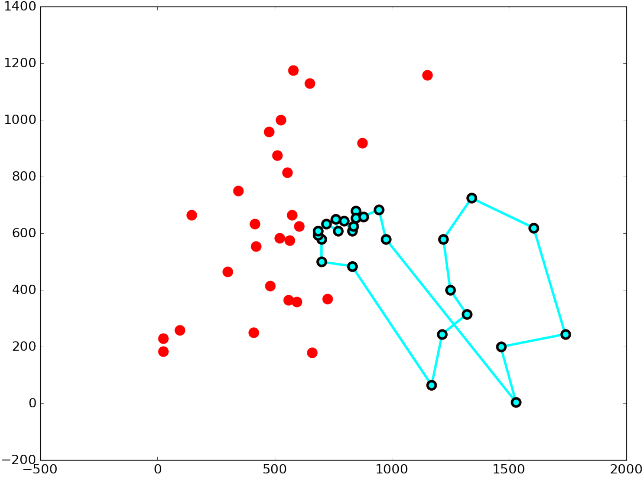
animation of the closest-neighbor algorithm



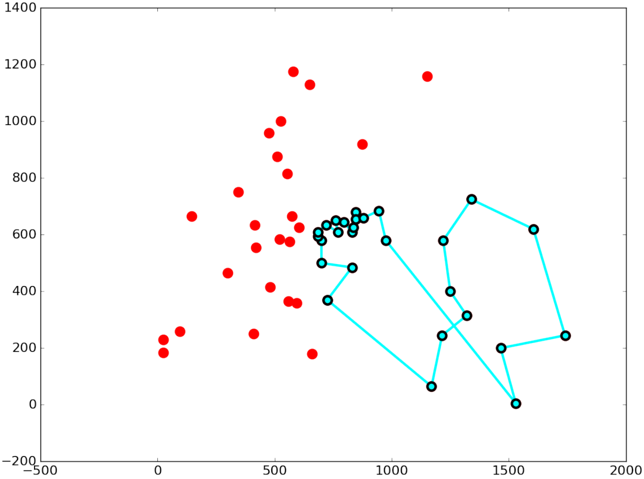
animation of the closest-neighbor algorithm



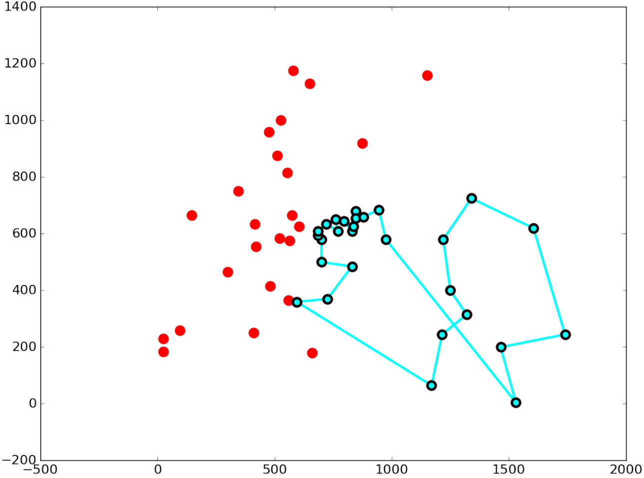
animation of the closest-neighbor algorithm



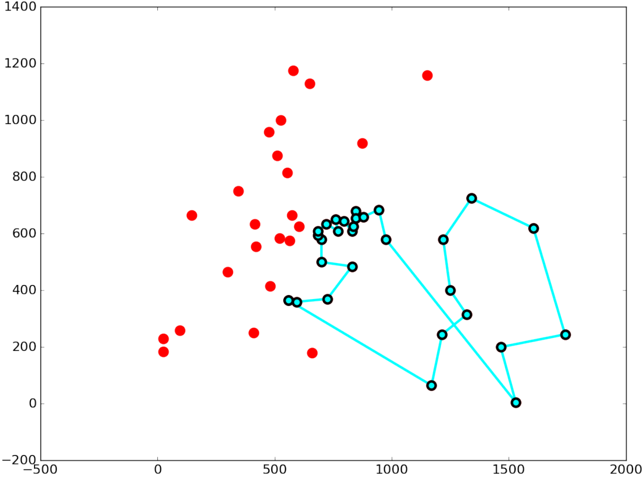
animation of the closest-neighbor algorithm



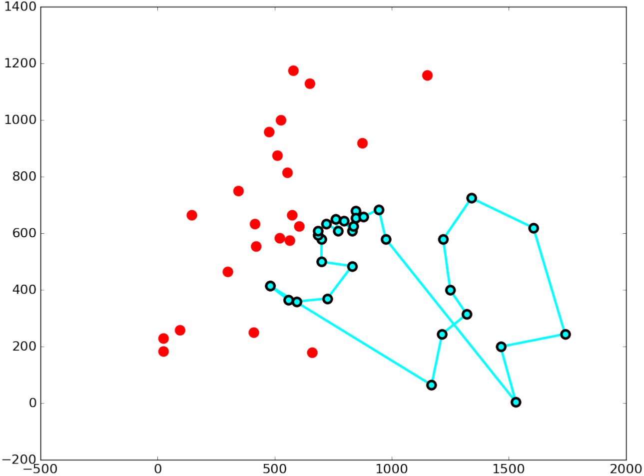
animation of the closest-neighbor algorithm



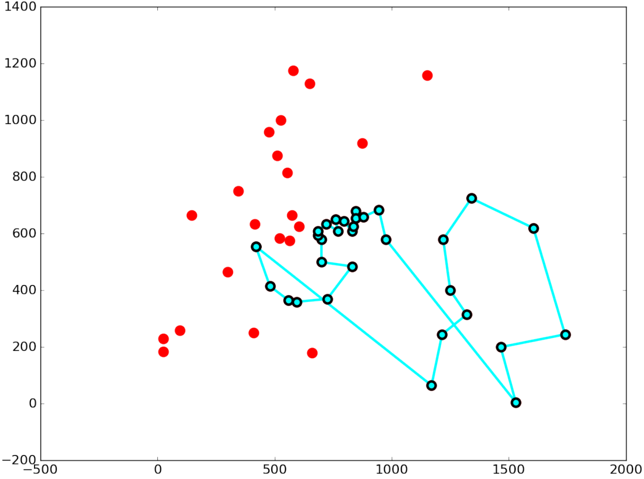
animation of the closest-neighbor algorithm



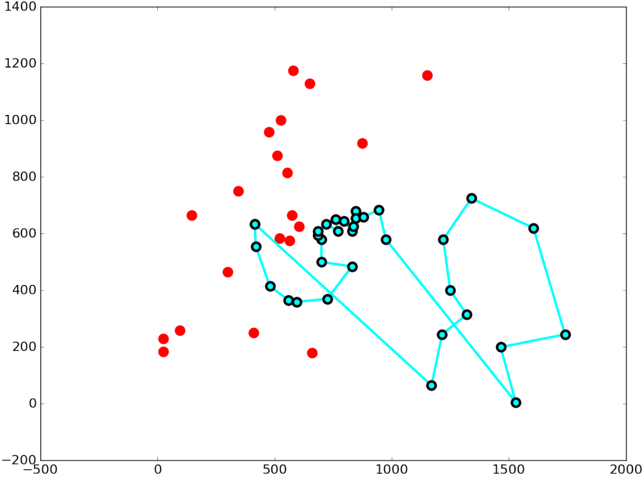
animation of the closest-neighbor algorithm



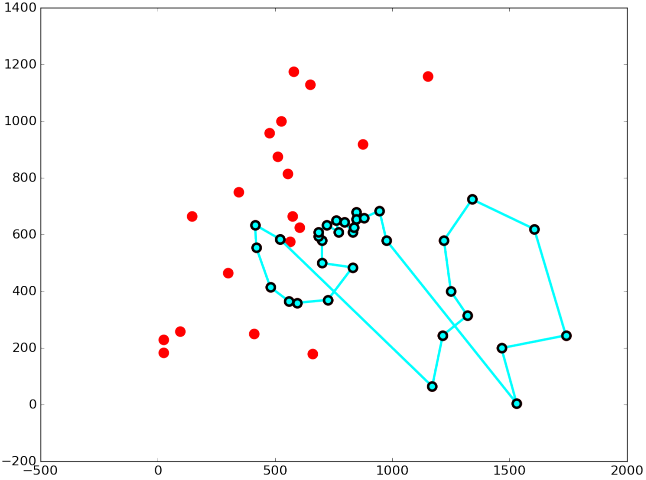
animation of the closest-neighbor algorithm



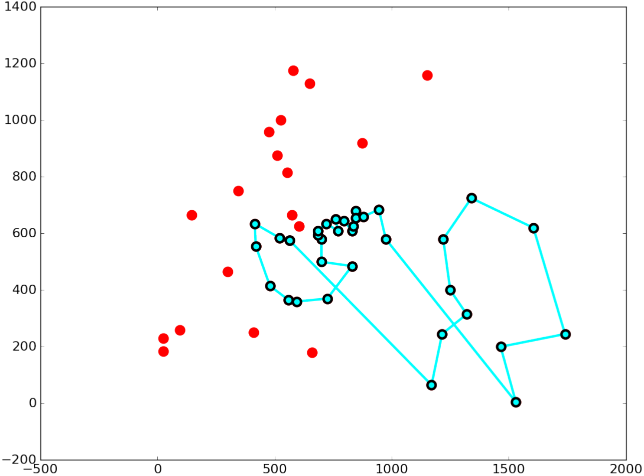
animation of the closest-neighbor algorithm



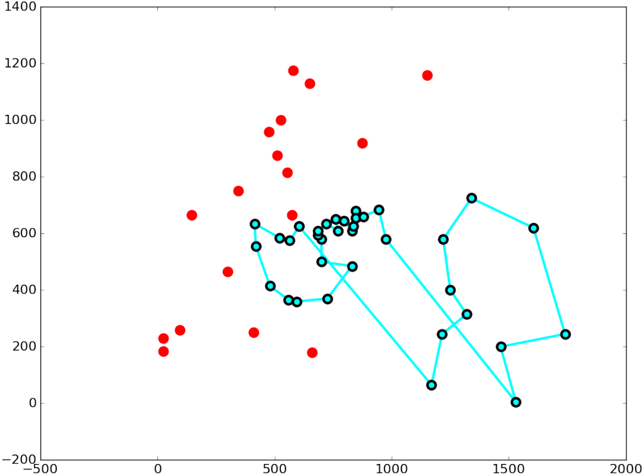
animation of the closest-neighbor algorithm



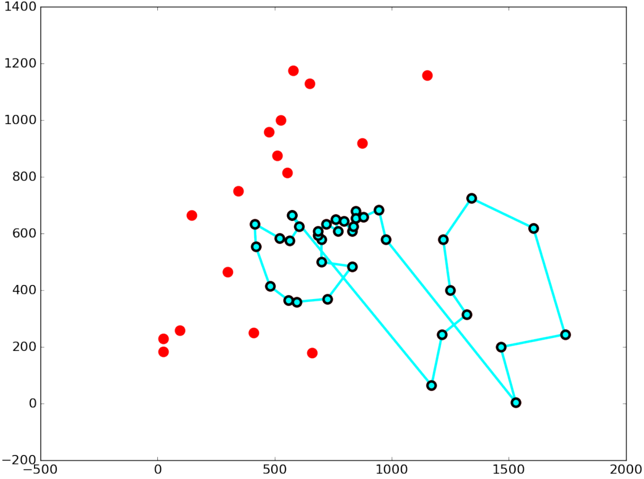
animation of the closest-neighbor algorithm



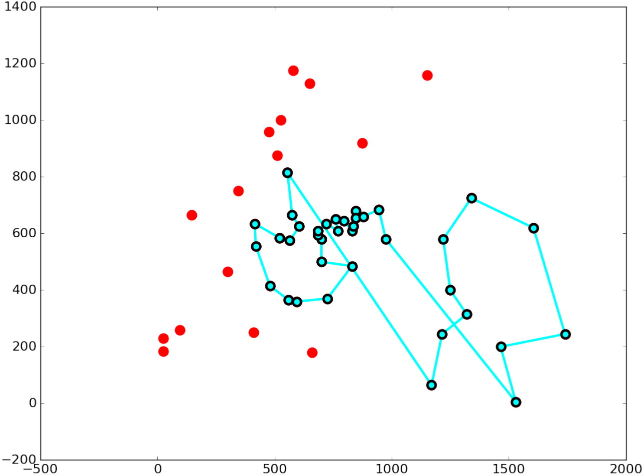
animation of the closest-neighbor algorithm



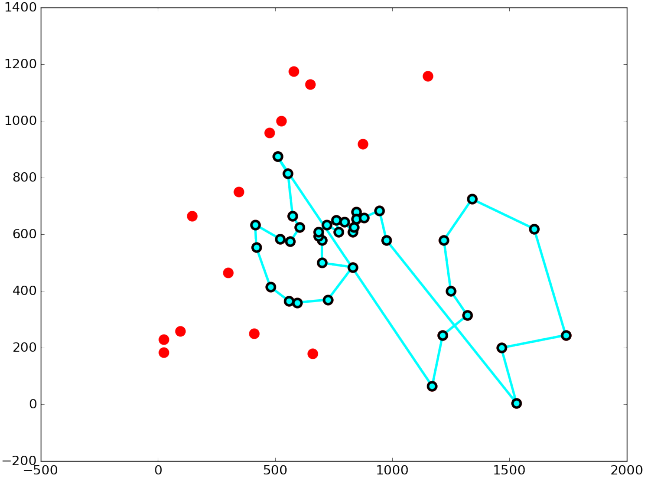
animation of the closest-neighbor algorithm



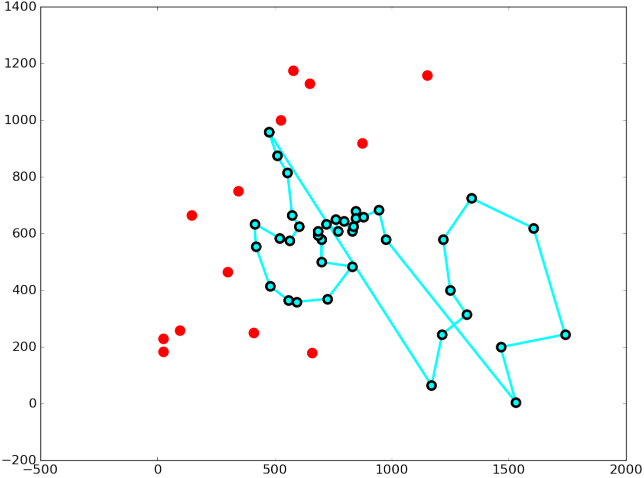
animation of the closest-neighbor algorithm



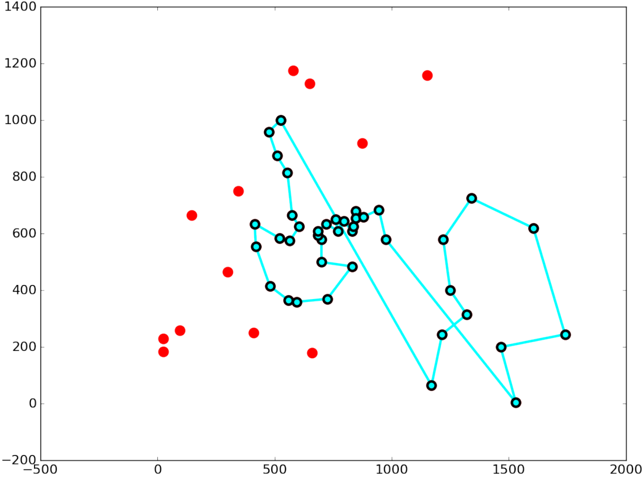
animation of the closest-neighbor algorithm



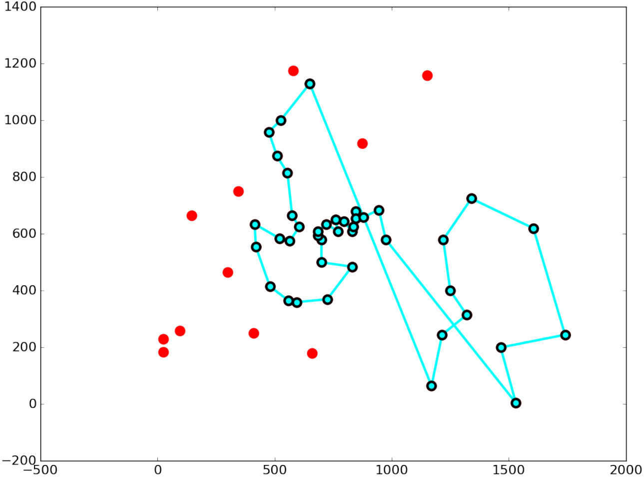
animation of the closest-neighbor algorithm



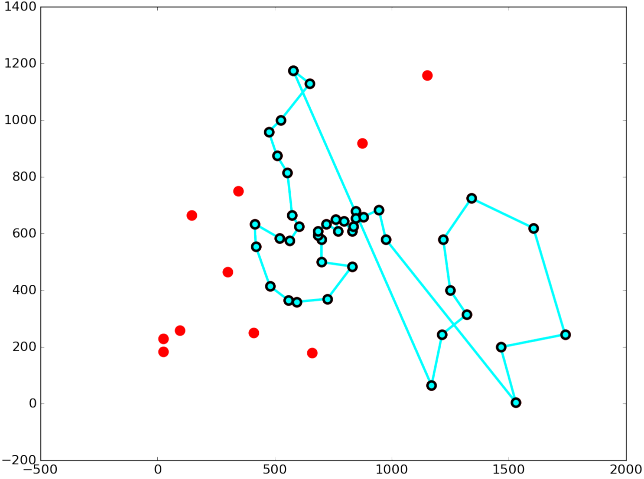
animation of the closest-neighbor algorithm



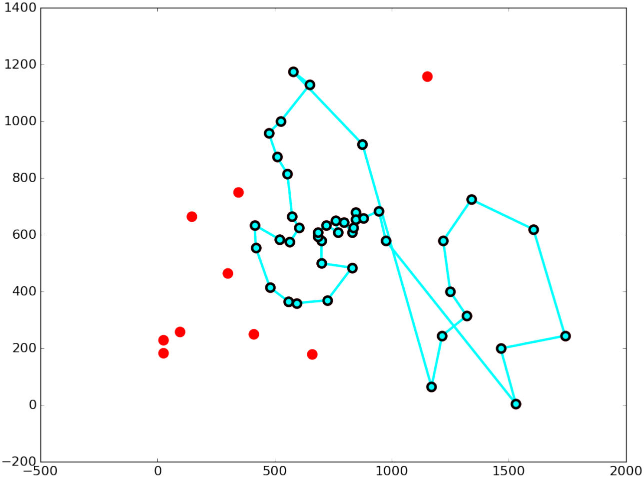
animation of the closest-neighbor algorithm



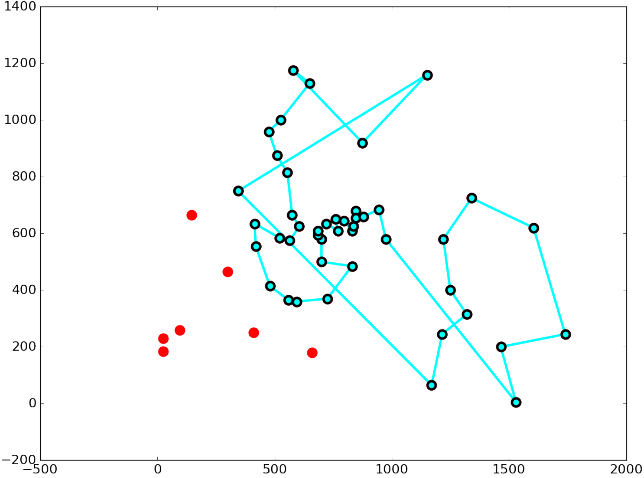
animation of the closest-neighbor algorithm



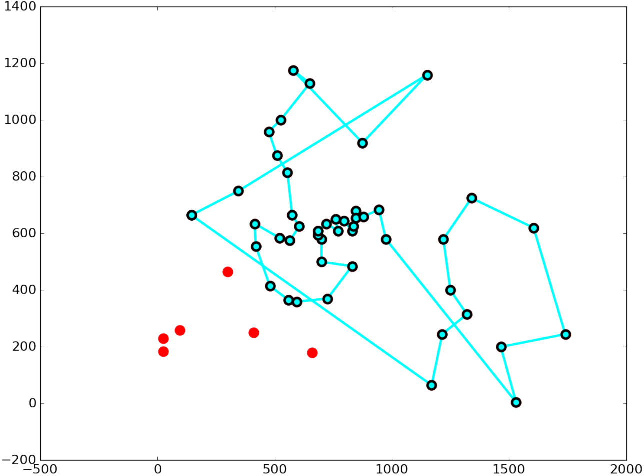
animation of the closest-neighbor algorithm

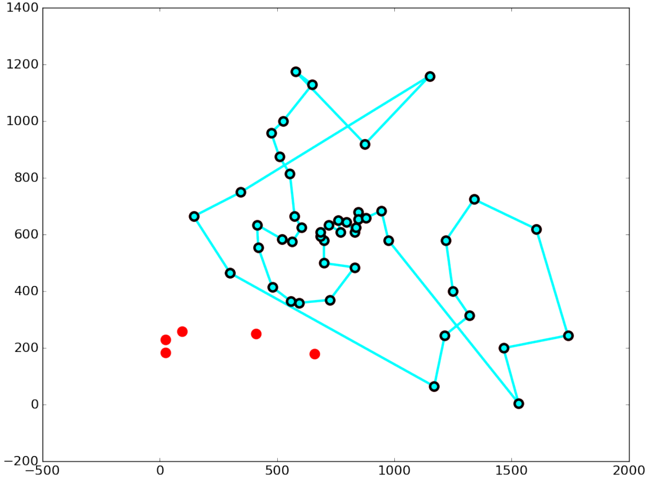


animation of the closest-neighbor algorithm

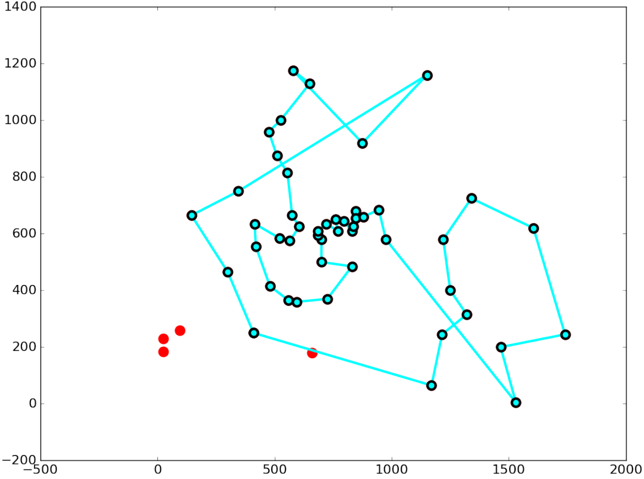


animation of the closest-neighbor algorithm

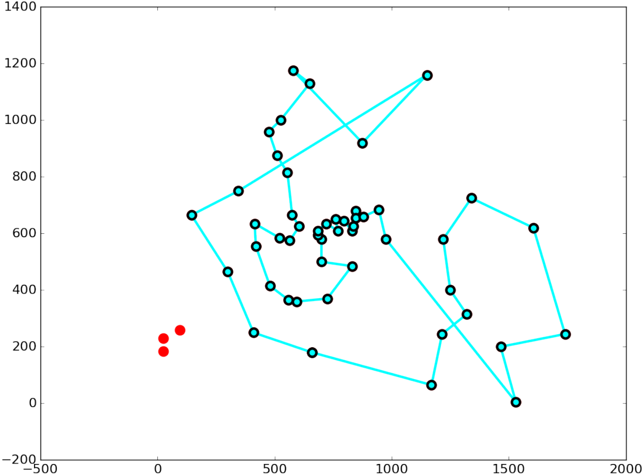



1400
1200
1000
800
600
400
200
0
−200
−500
0
500
1000
1500
2000

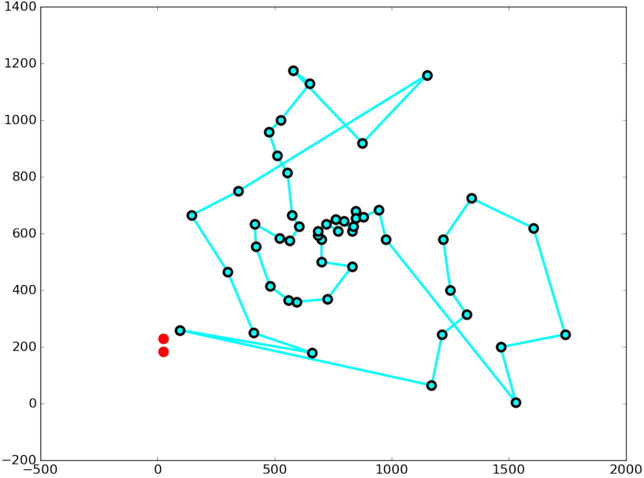
animation of the closest-neighbor algorithm



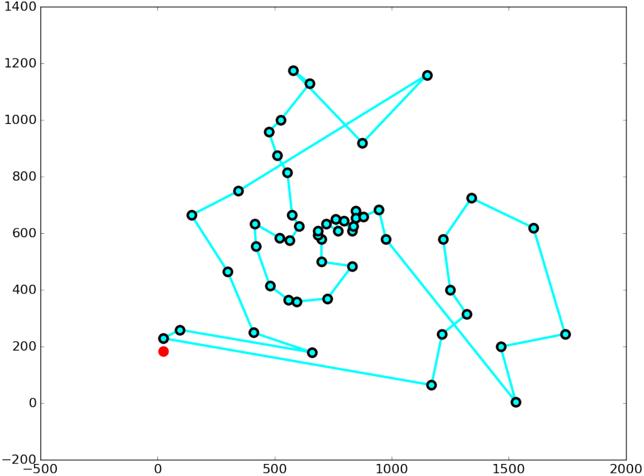
animation of the closest-neighbor algorithm



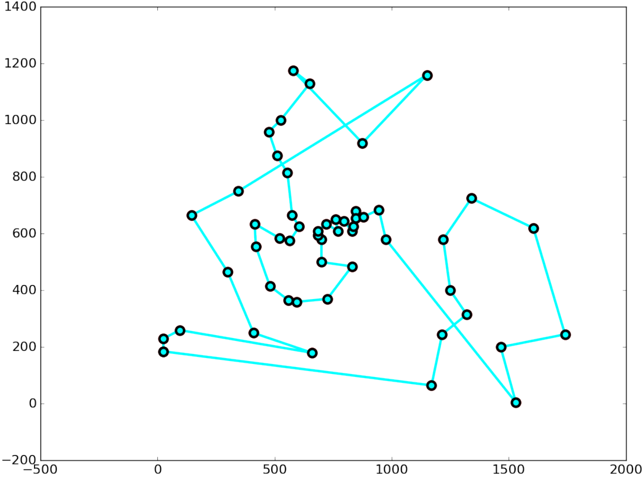
animation of the closest-neighbor algorithm



animation of the closest-neighbor algorithm



animation of the closest-neighbor algorithm



How does the quick tour algorithm work?

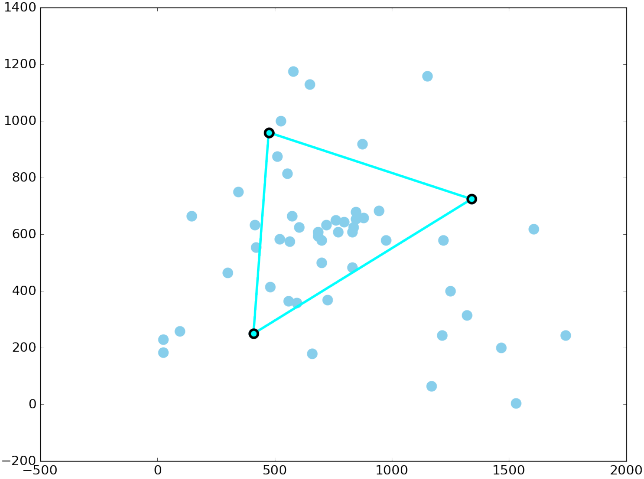
The quick tour algorithm is my own contribution (but found to be known as *insertion/addition approach*) for this course. It is a probabilistic greedy algorithm:

- select three random cities to form an initial triangular tour
- while there are still unconnected cities
  - choose a random (closest) unconnected city
  - expand the current tour by inserting the new city such that the tour increment is minimal

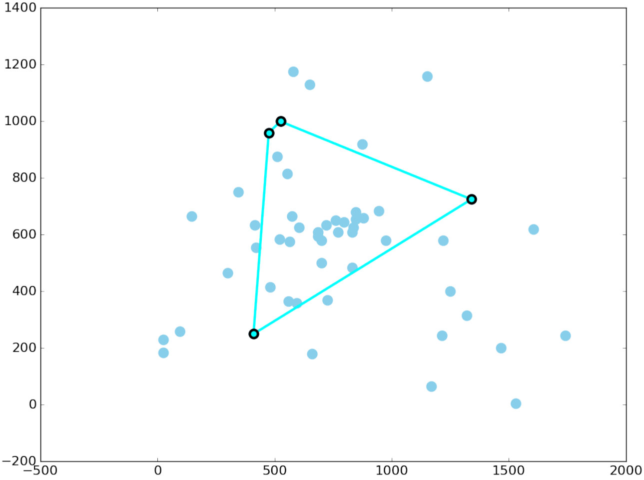
- runtime is in $\mathcal{O}(n^2)$,
- the random version can be run in Monte Carlo fashion keeping the shortest tour
- worst tour may have a length up to $2 \cdot L_{opt}$

There are more similar approaches, e.g., nearest addition or farthest addition.

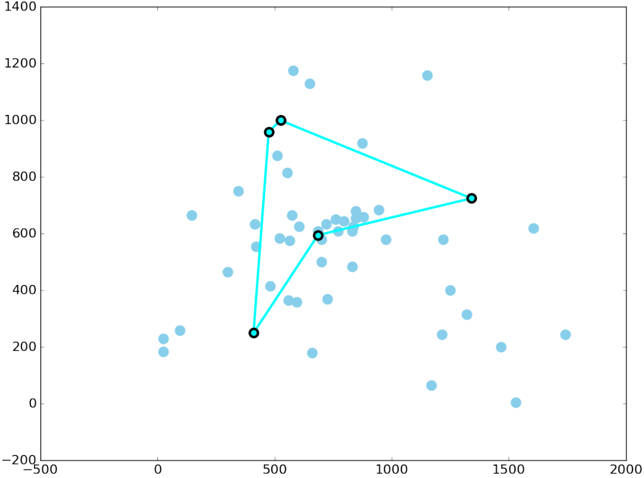
animation of the quick tour algorithm



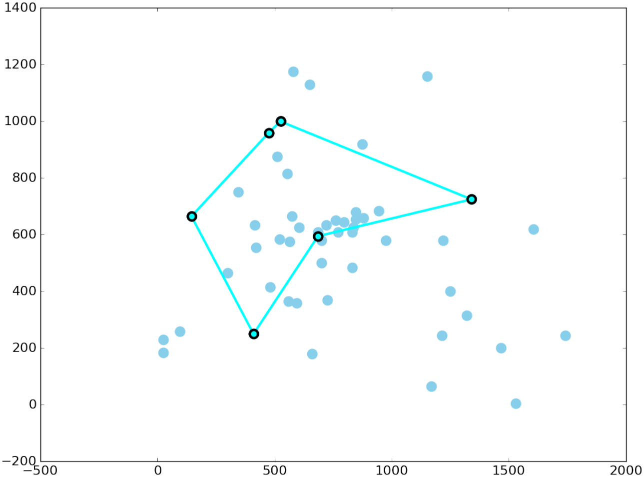
animation of the quick tour algorithm



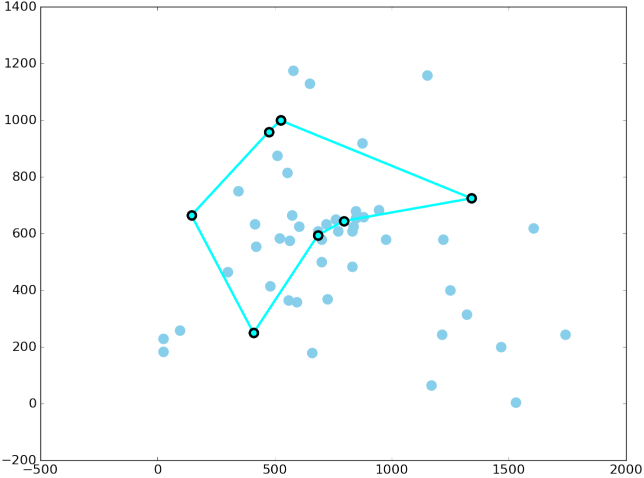
animation of the quick tour algorithm



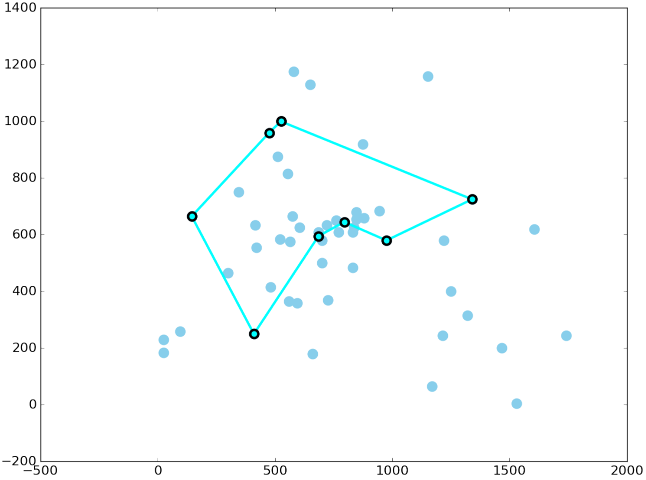
animation of the quick tour algorithm

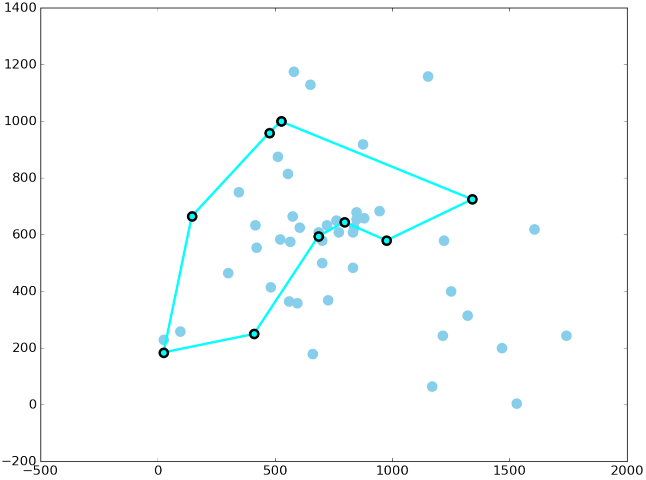


animation of the quick tour algorithm

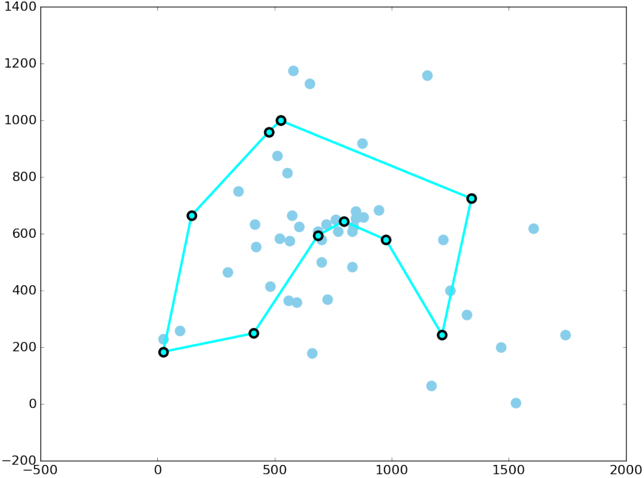


animation of the quick tour algorithm

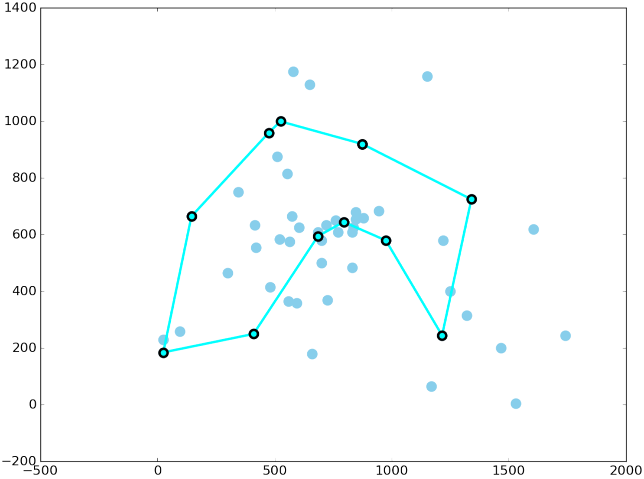



1400
1200
1000
800
600
400
200
0
−200
−500
0
500
1000
1500
2000

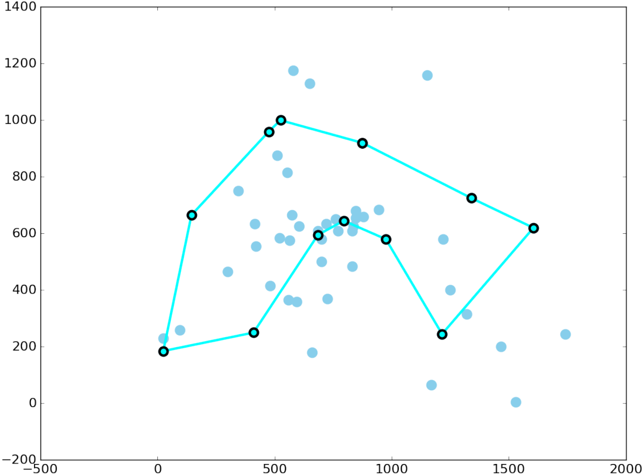
animation of the quick tour algorithm



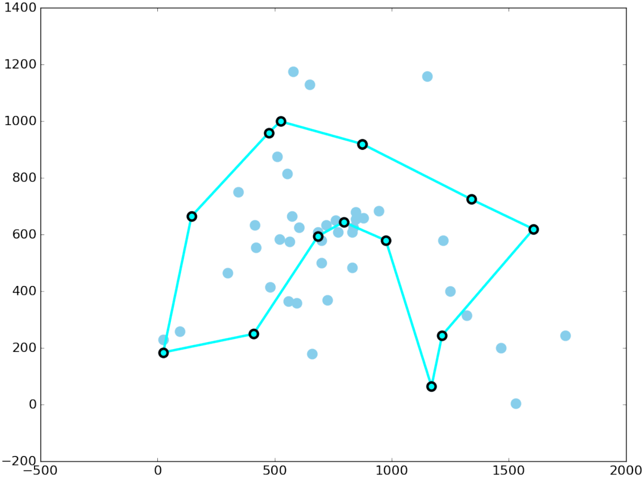
animation of the quick tour algorithm



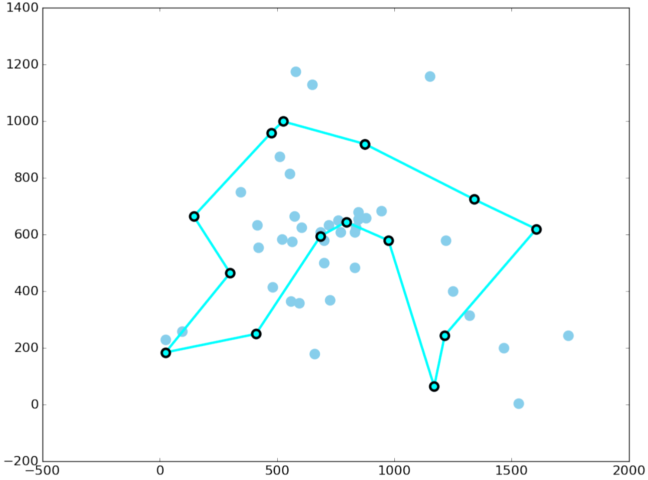
animation of the quick tour algorithm



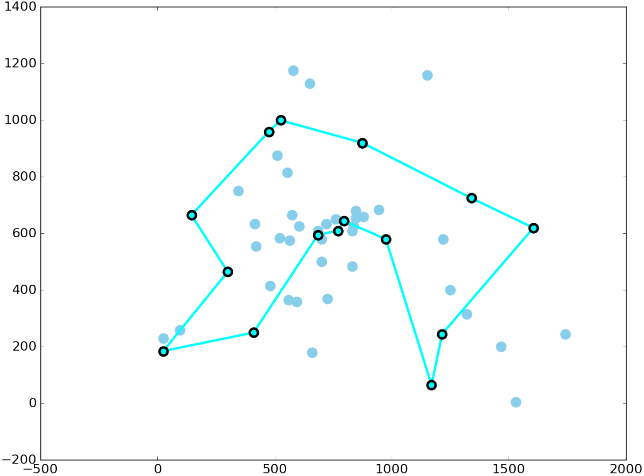
animation of the quick tour algorithm



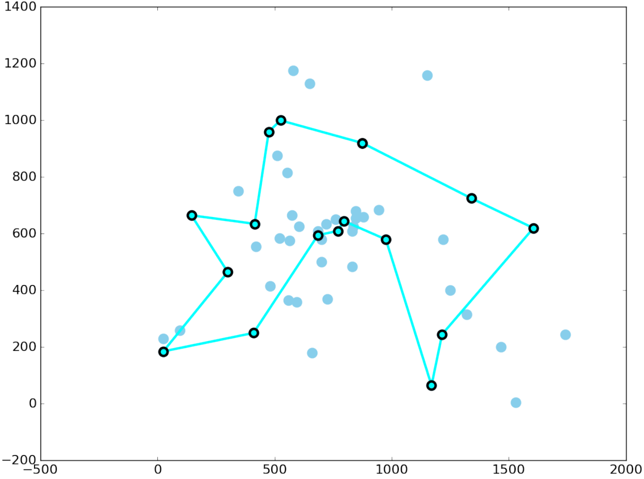
animation of the quick tour algorithm



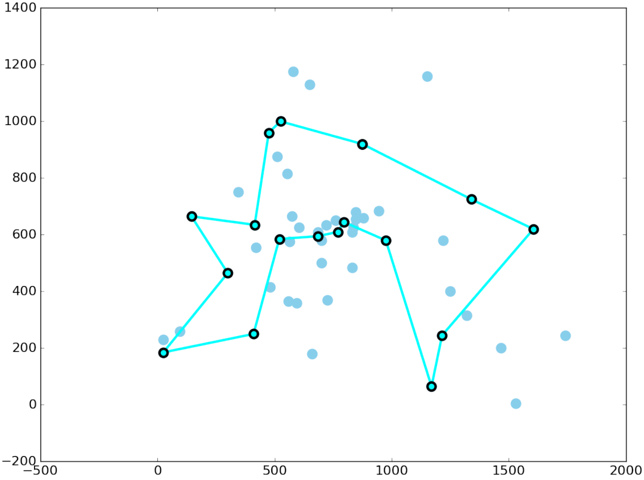
animation of the quick tour algorithm



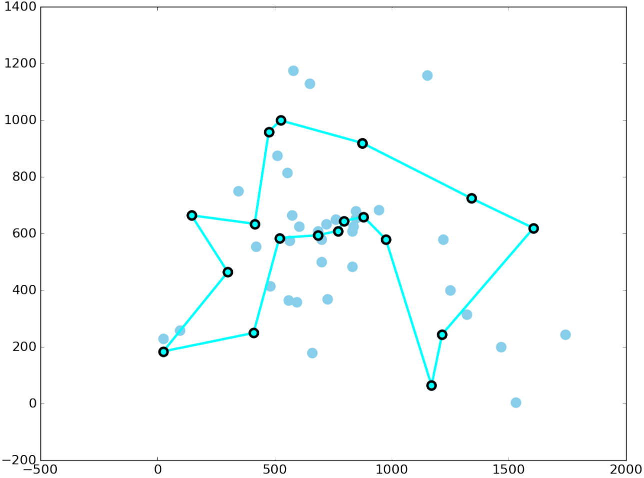
animation of the quick tour algorithm



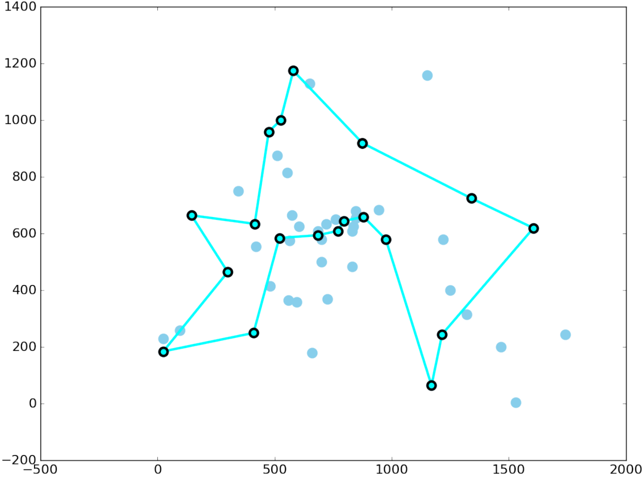
animation of the quick tour algorithm



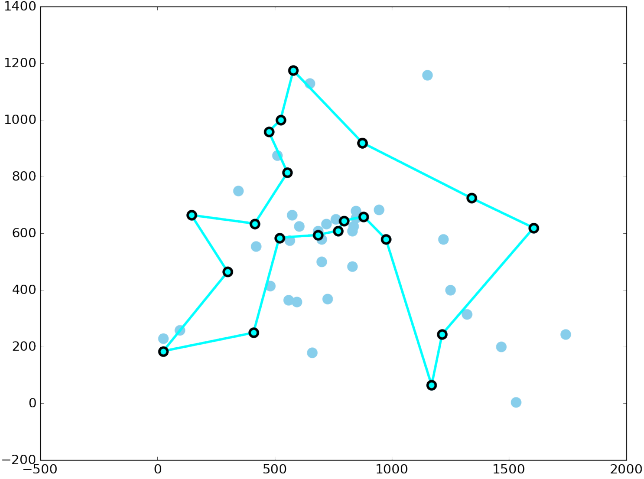
animation of the quick tour algorithm



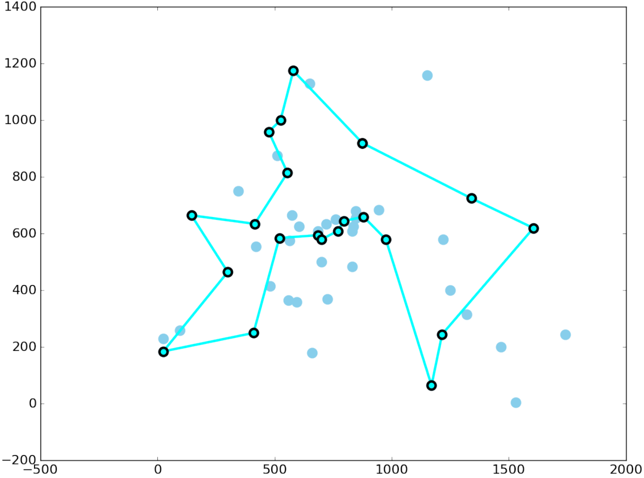
animation of the quick tour algorithm



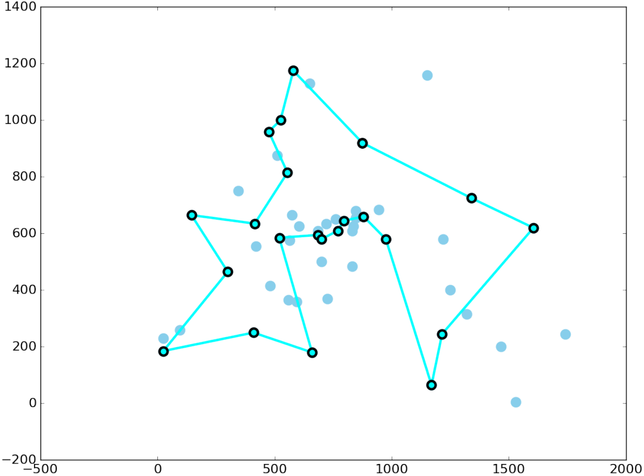
animation of the quick tour algorithm



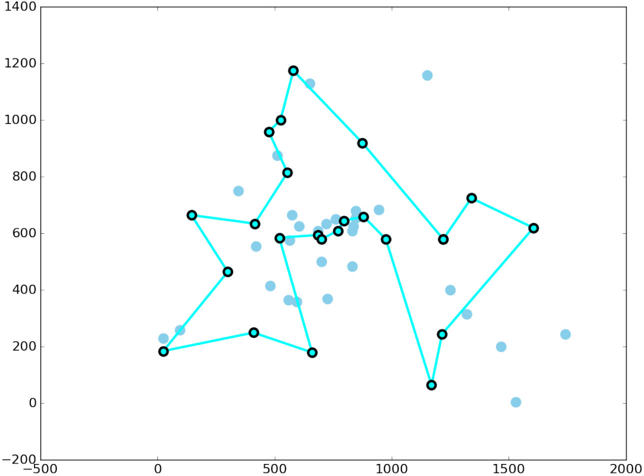
animation of the quick tour algorithm



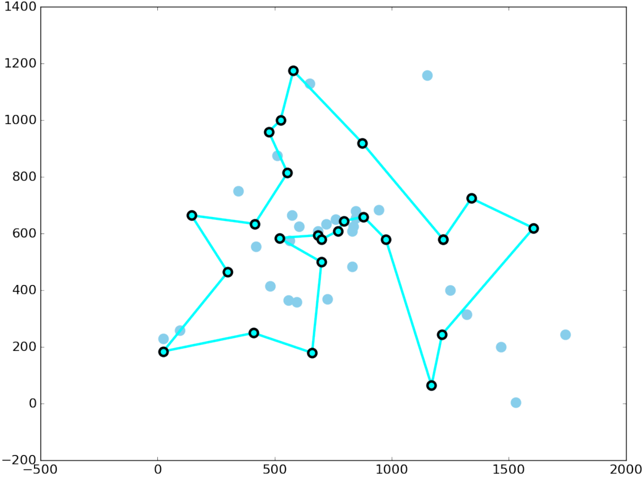
animation of the quick tour algorithm

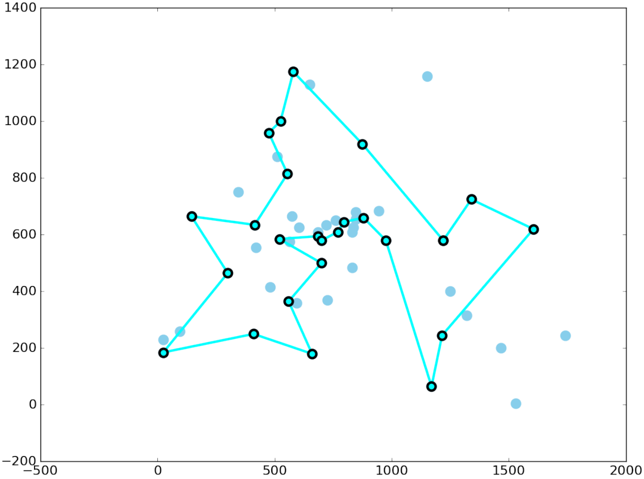


animation of the quick tour algorithm

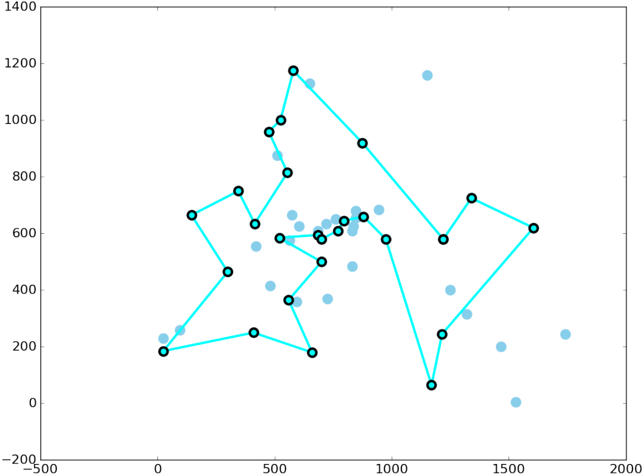


animation of the quick tour algorithm

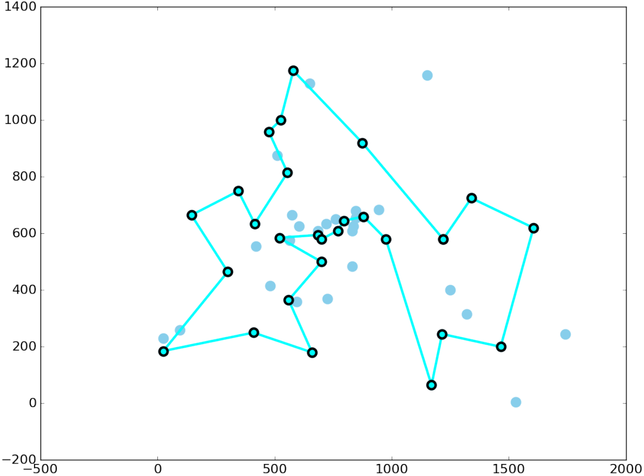



1400
1200
1000
800
600
400
200
0
−200
−500
0
500
1000
1500
2000

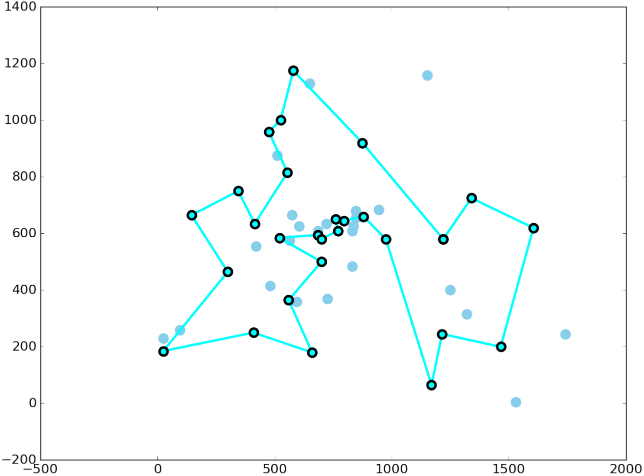
animation of the quick tour algorithm



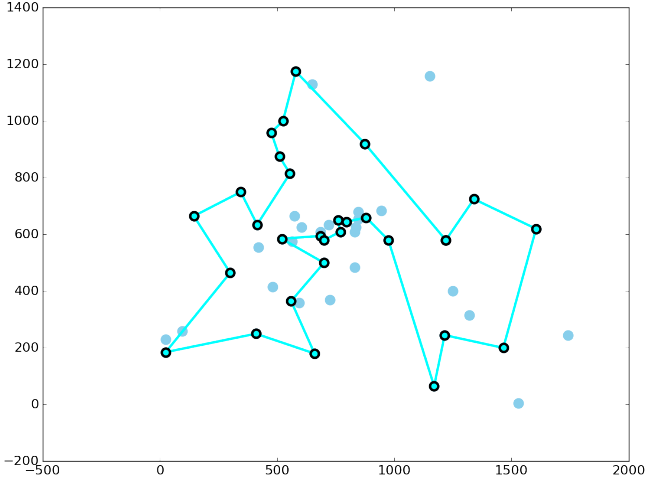
animation of the quick tour algorithm



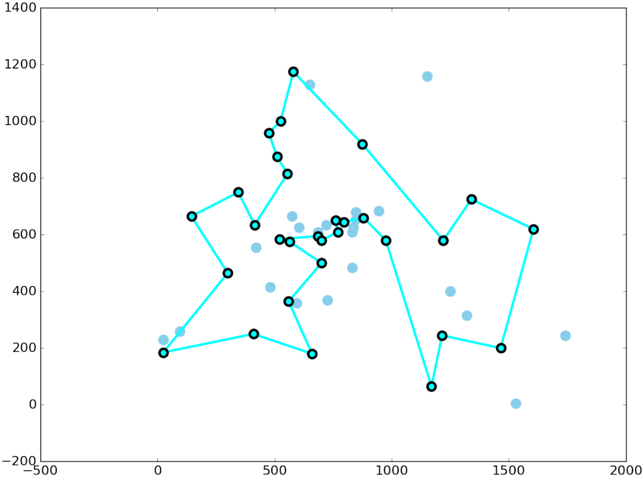
animation of the quick tour algorithm



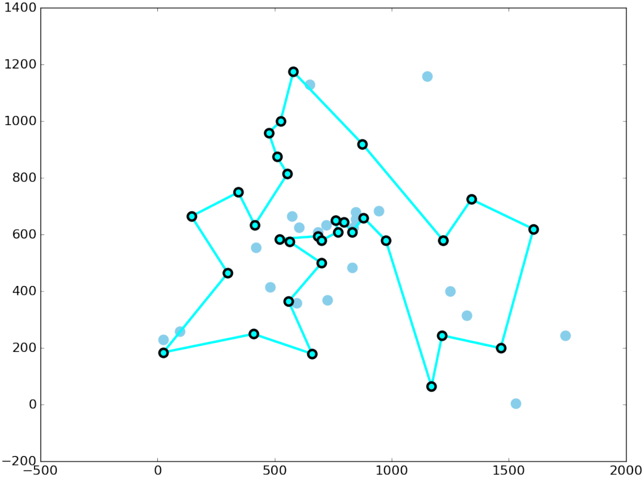
animation of the quick tour algorithm



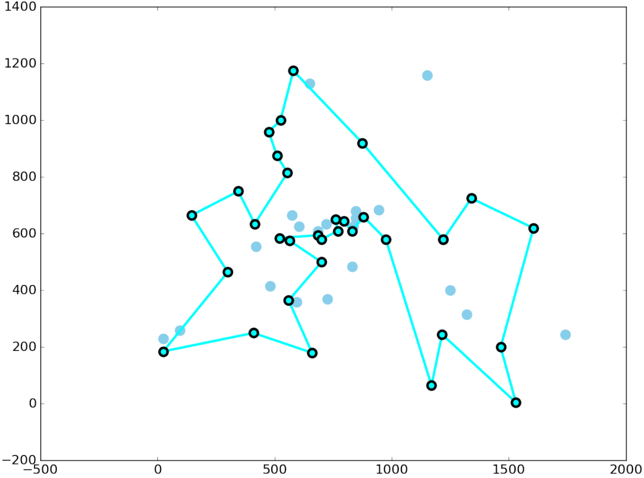
animation of the quick tour algorithm



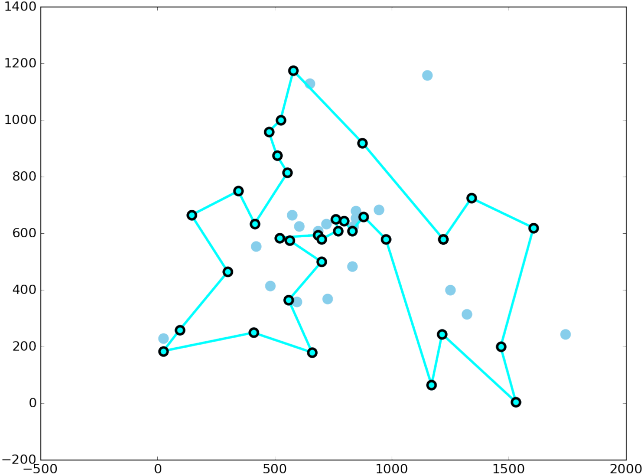
animation of the quick tour algorithm



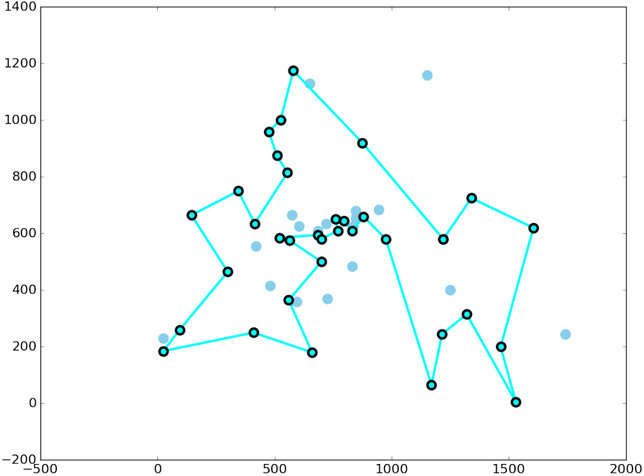
animation of the quick tour algorithm



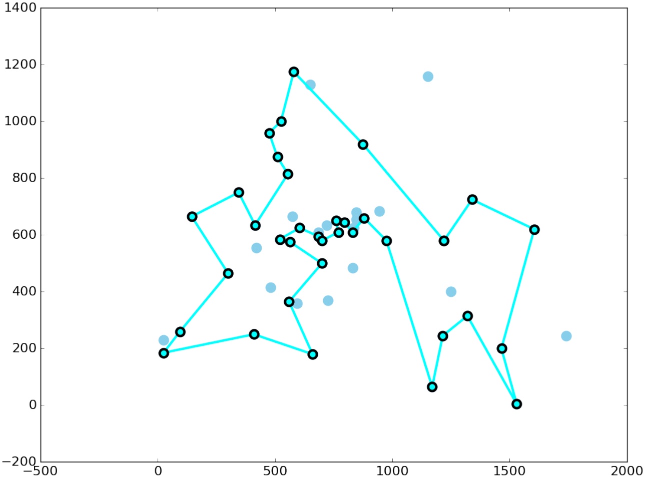
animation of the quick tour algorithm



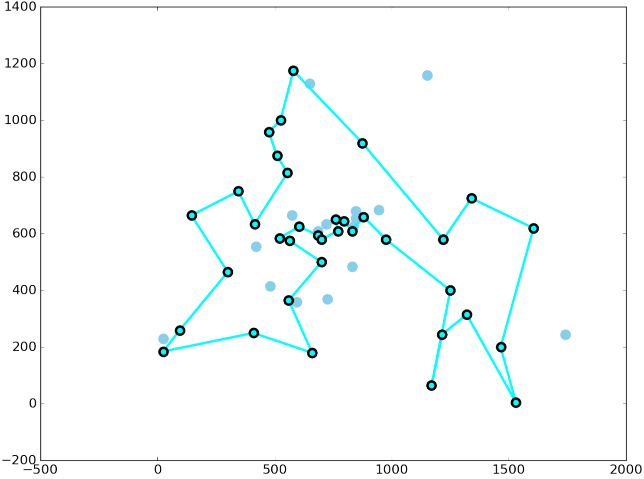
animation of the quick tour algorithm



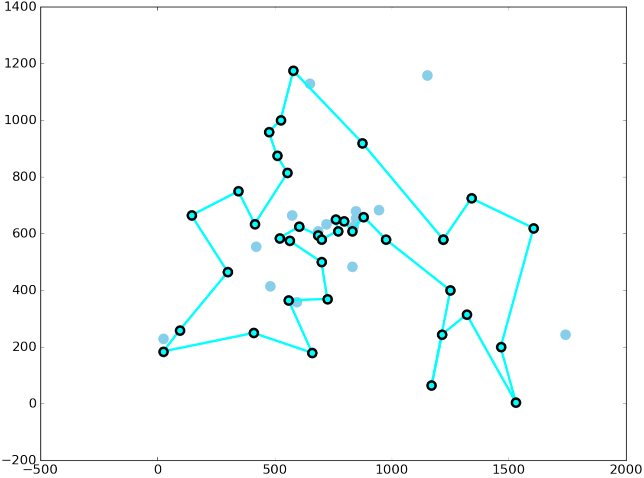
animation of the quick tour algorithm



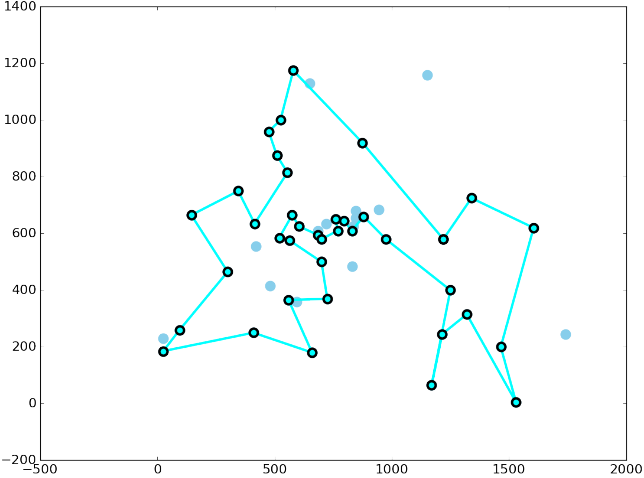
animation of the quick tour algorithm



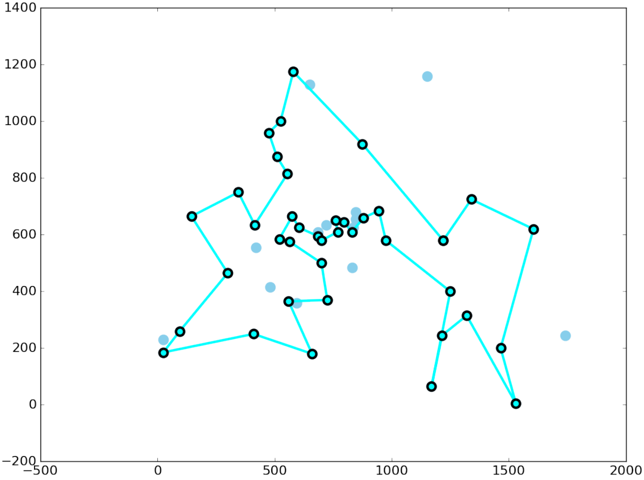
animation of the quick tour algorithm



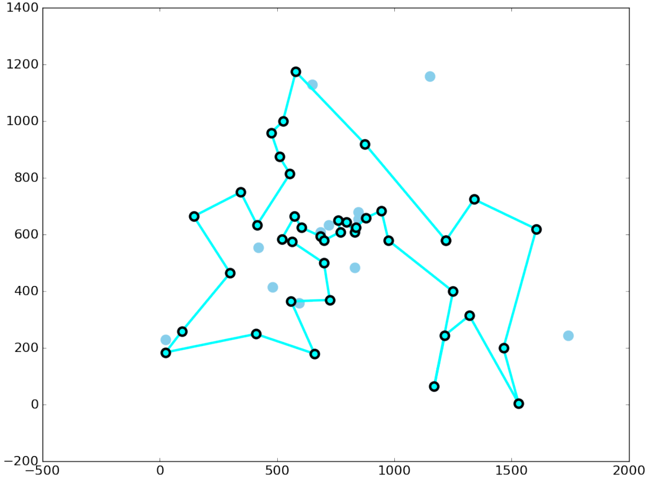
animation of the quick tour algorithm



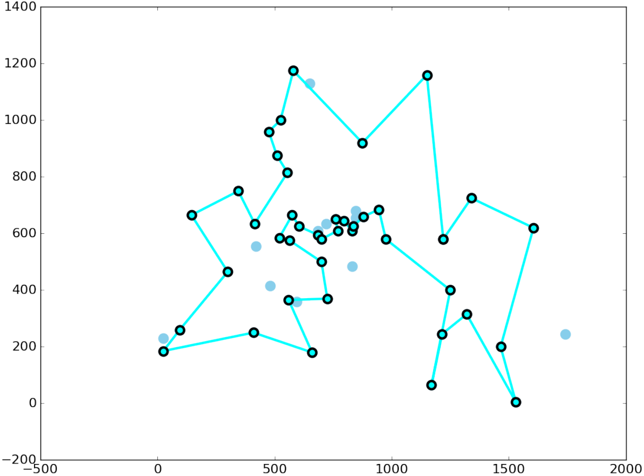
animation of the quick tour algorithm



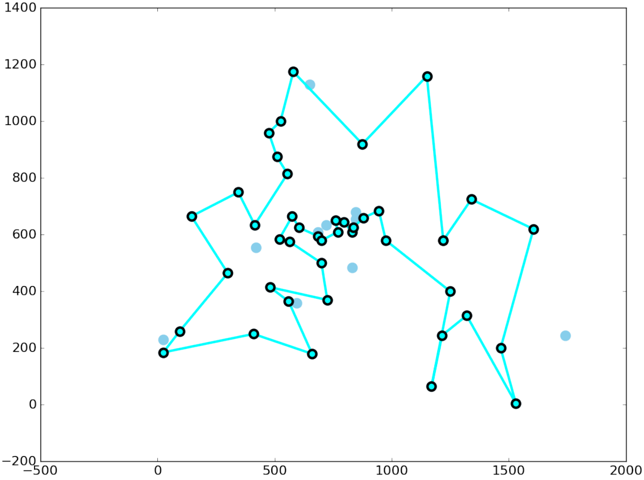
animation of the quick tour algorithm

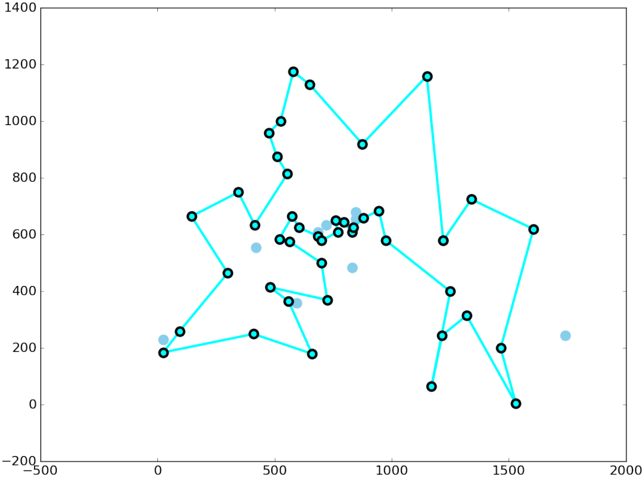


animation of the quick tour algorithm

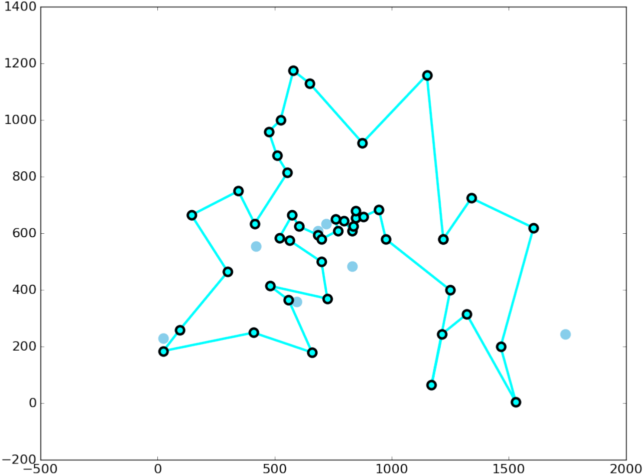


animation of the quick tour algorithm

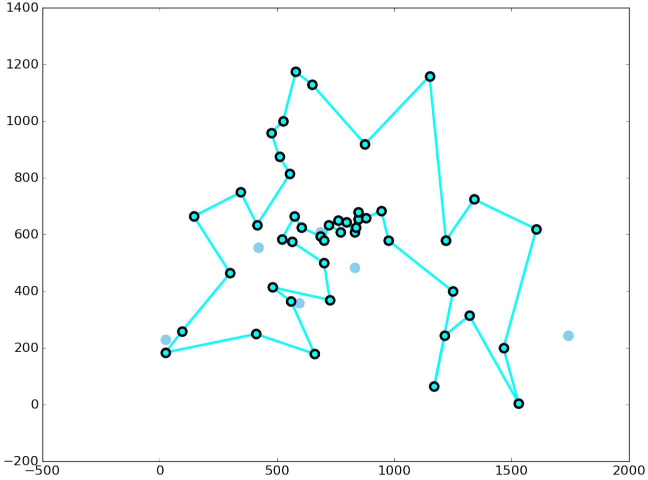



1400
1200
1000
800
600
400
200
0
−200
−500
0
500
1000
1500
2000

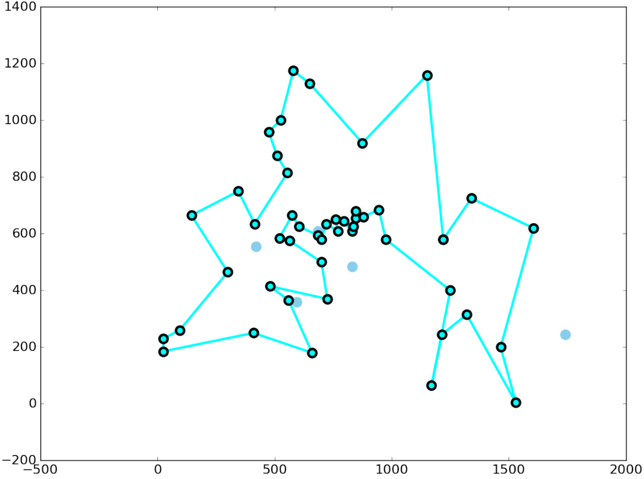
animation of the quick tour algorithm



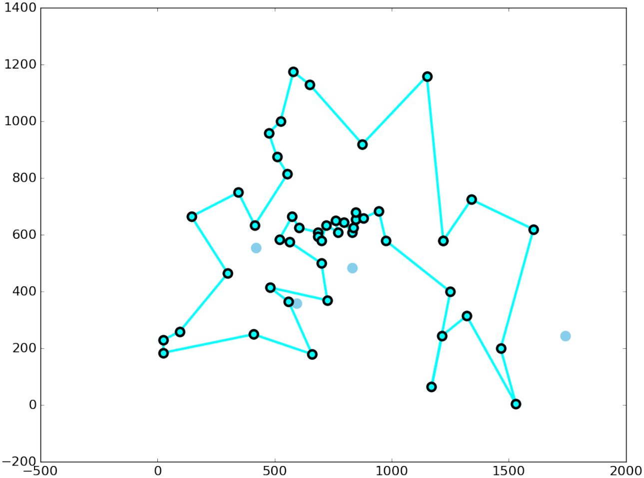
animation of the quick tour algorithm



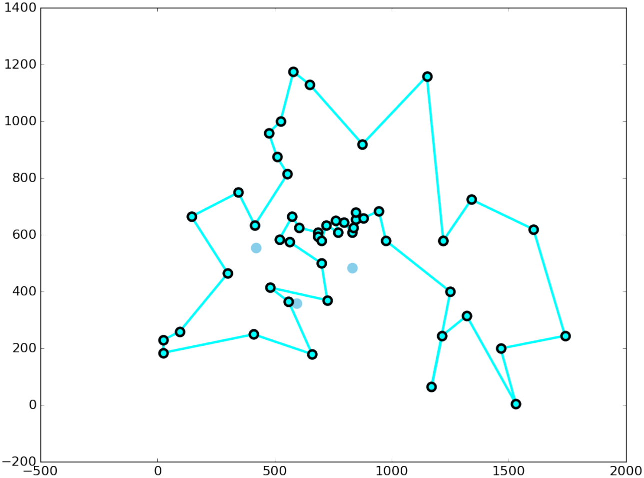
animation of the quick tour algorithm



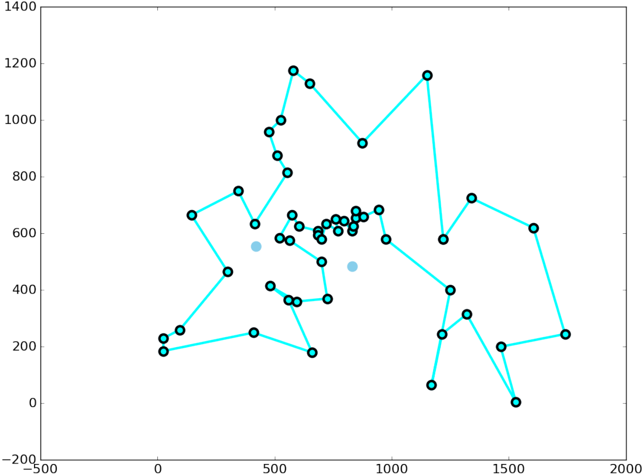
animation of the quick tour algorithm



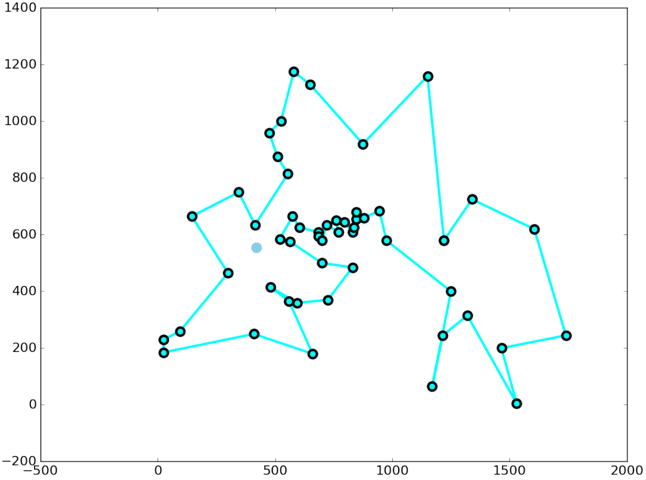
animation of the quick tour algorithm



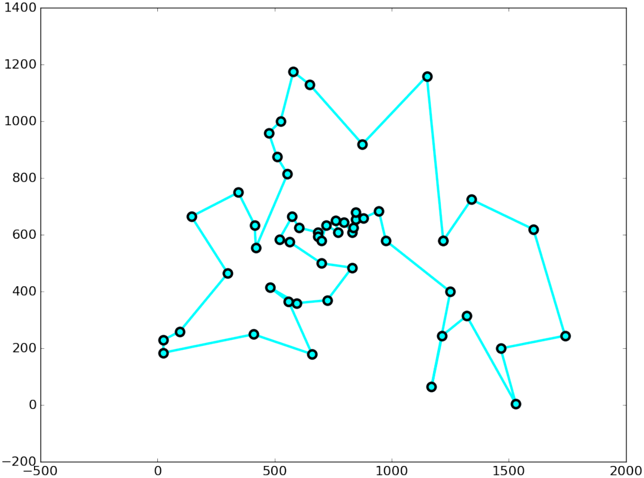
animation of the quick tour algorithm



animation of the quick tour algorithm



animation of the quick tour algorithm



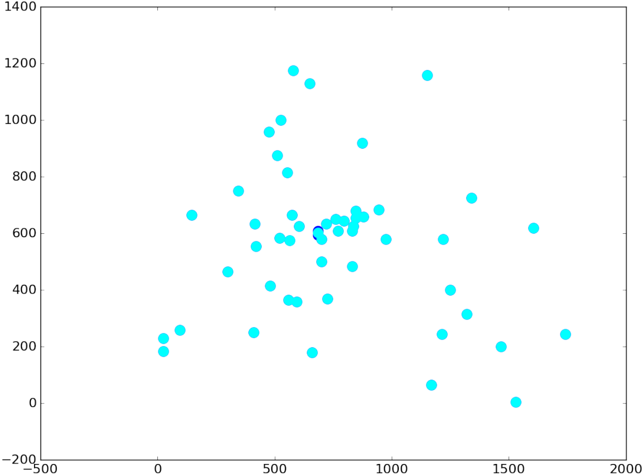
How does the pair-center algorithm work?

The pair-center tour algorithm is a contribution of my own for this course (still haven't found it on the internet). It is a deterministic algorithm:

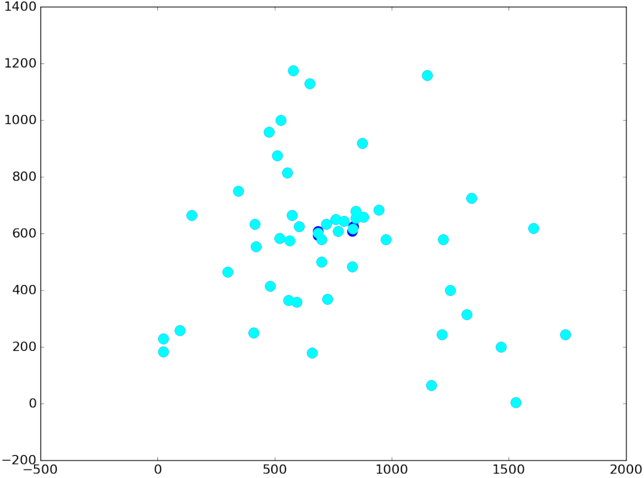
- with a bottom-up construction build a binary tree by replacing the/a closest pair of points by their center
- with a top-down construction build the tour by inserting the corresponding pairs in the best possible way

The runtime is in $\mathcal{O}(n^2)$, I guess (implemented is $\mathcal{O}(n^3)$ and a $\mathcal{O}(n^2)$ version with more sophisticated data structures).
Can you prove a worst case bound for the tour length? Recently I've improved to $\mathcal{O}(n\mathrm{polylog}(n))$ with practical runtime in the order of $n \log n$ and error below 5%.

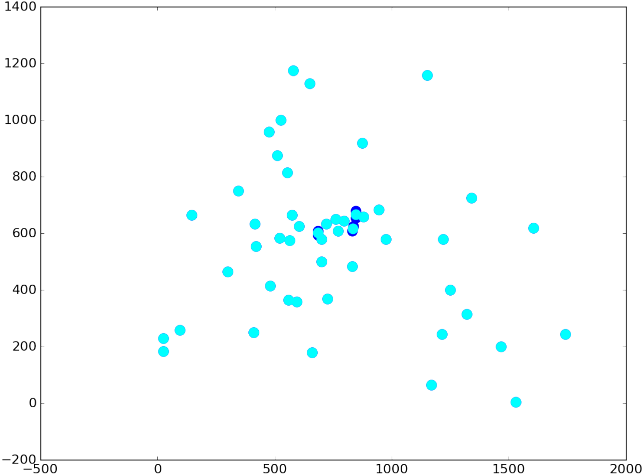
animation of the pair-center tour algorithm



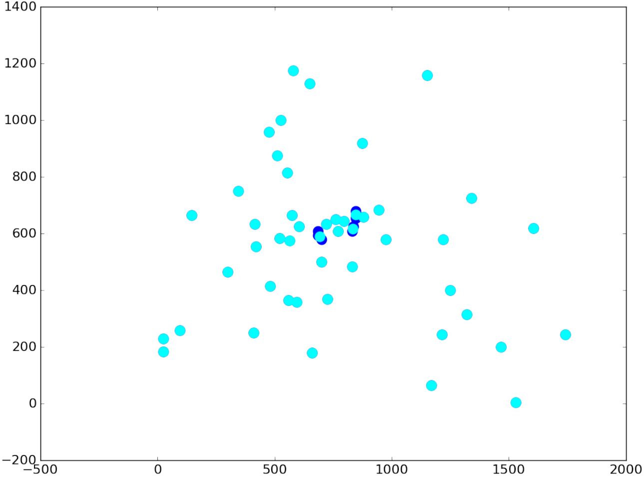
animation of the pair-center tour algorithm



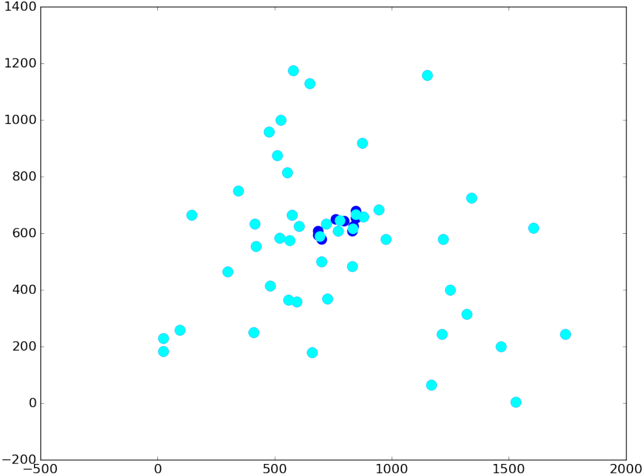
animation of the pair-center tour algorithm



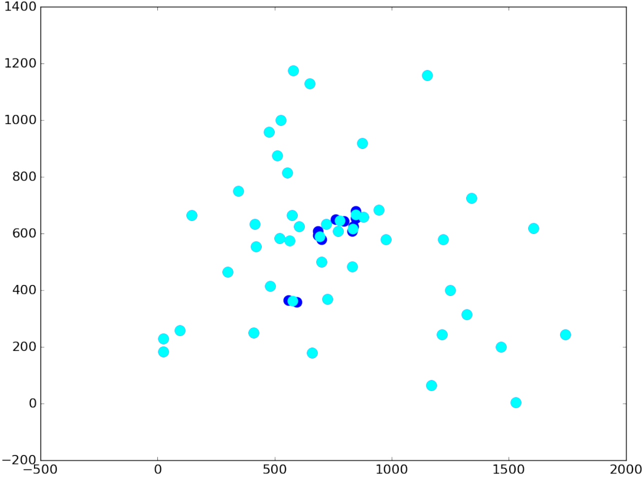
animation of the pair-center tour algorithm



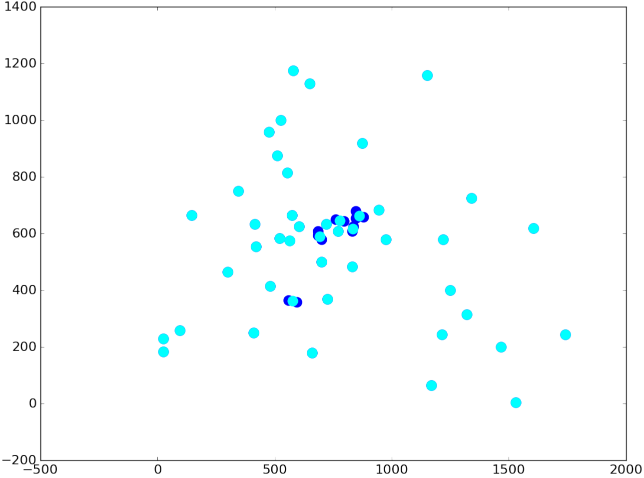
animation of the pair-center tour algorithm



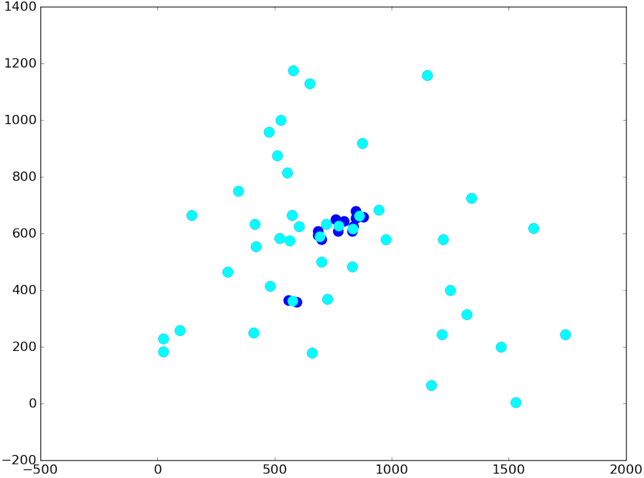
animation of the pair-center tour algorithm



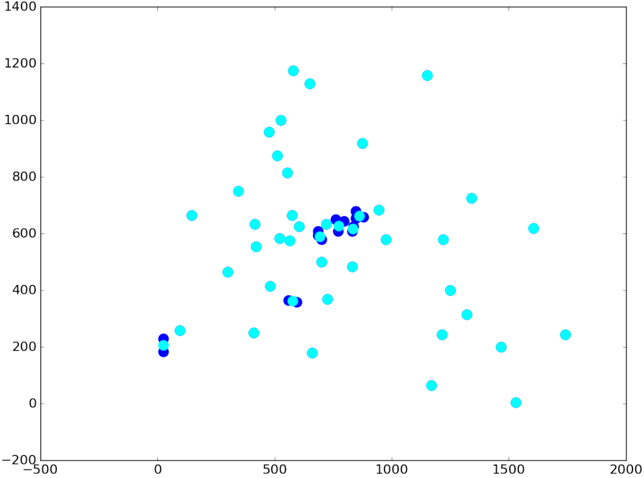
animation of the pair-center tour algorithm



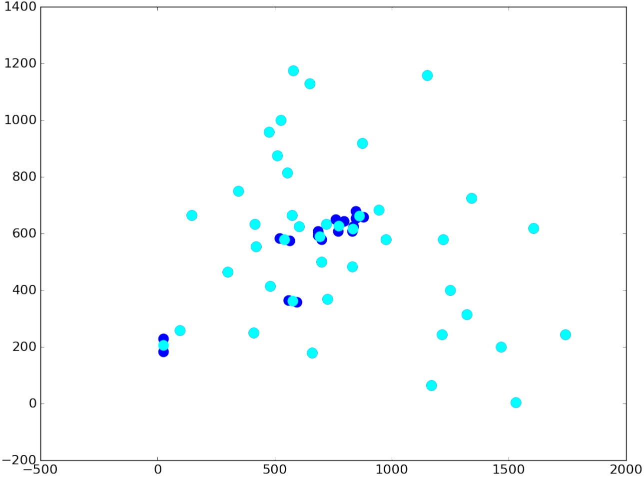
animation of the pair-center tour algorithm



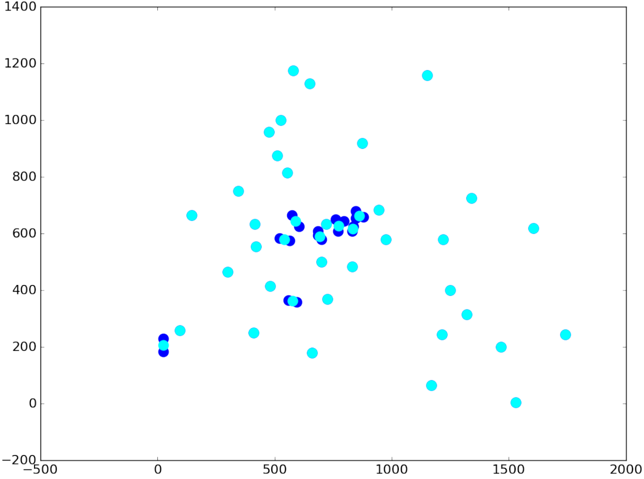
animation of the pair-center tour algorithm



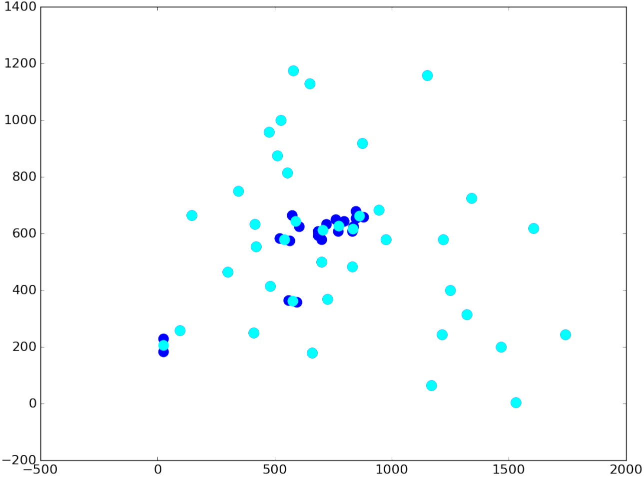
animation of the pair-center tour algorithm



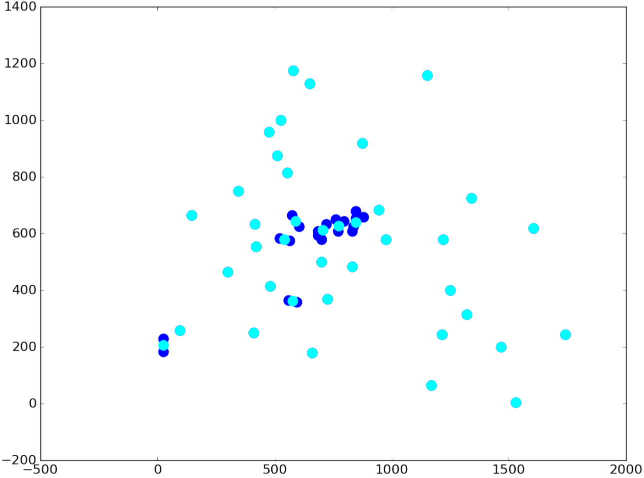
animation of the pair-center tour algorithm



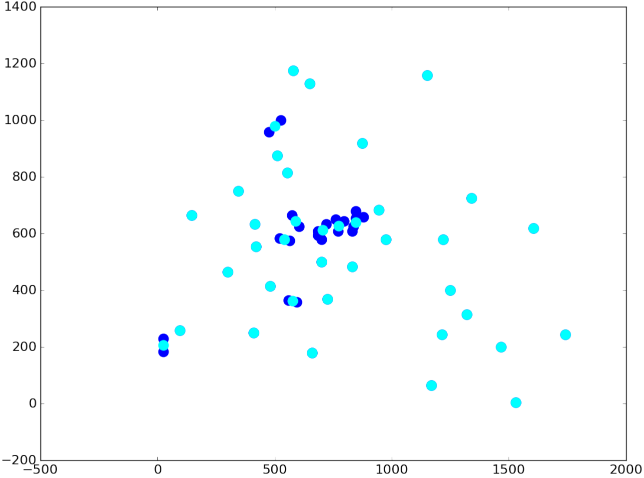
animation of the pair-center tour algorithm



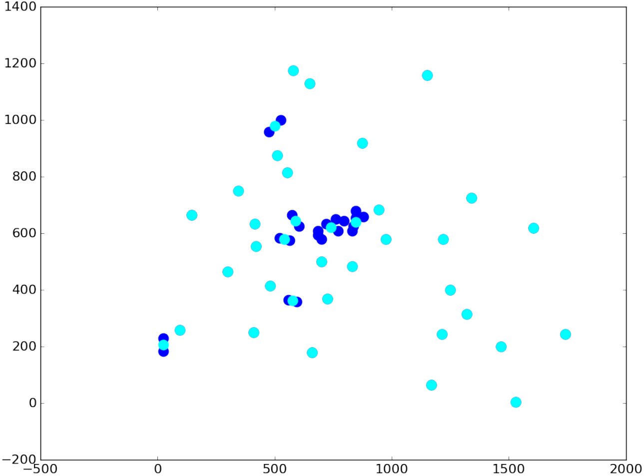
animation of the pair-center tour algorithm



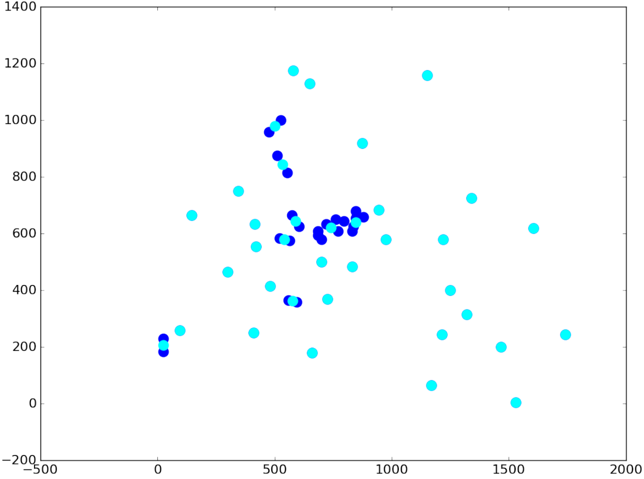
animation of the pair-center tour algorithm



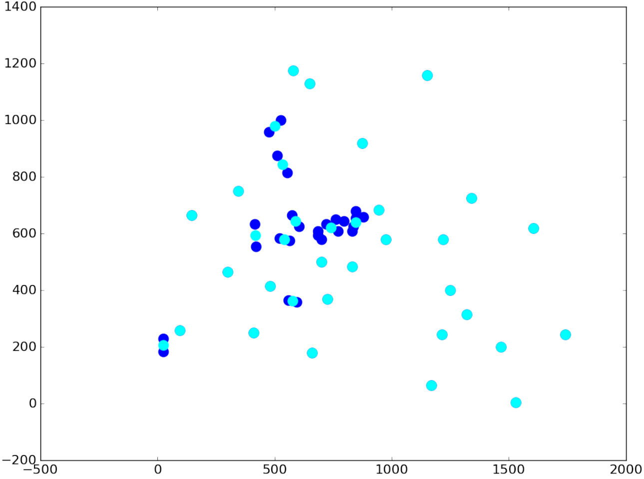
animation of the pair-center tour algorithm



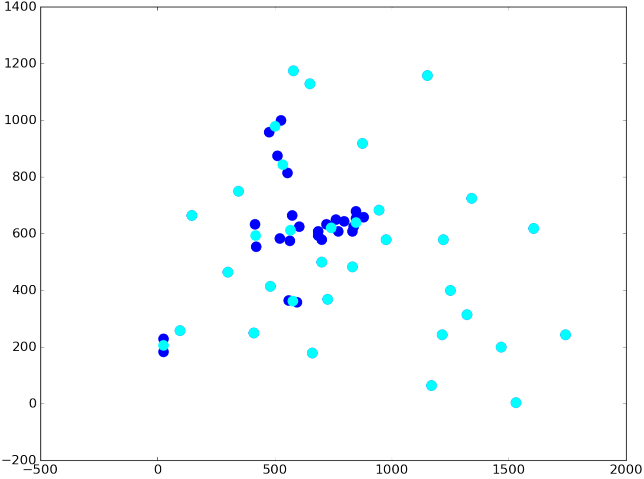
animation of the pair-center tour algorithm



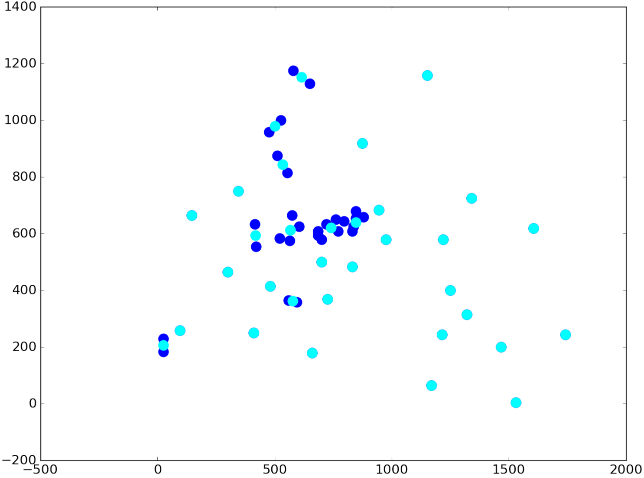
animation of the pair-center tour algorithm



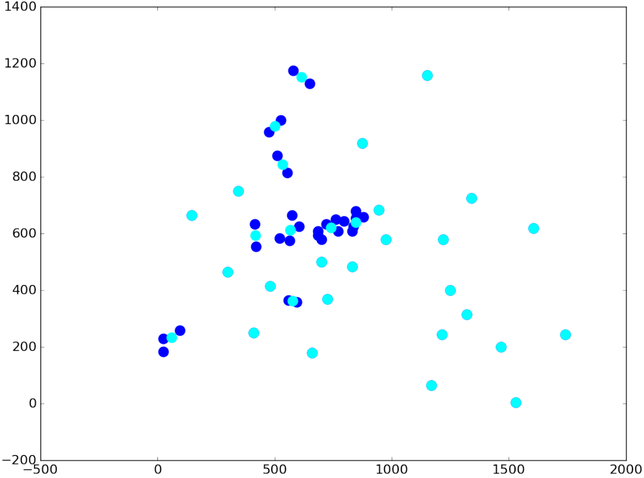
animation of the pair-center tour algorithm



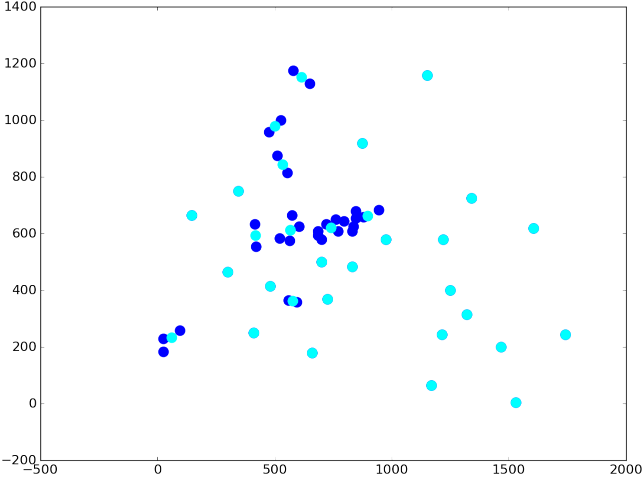
animation of the pair-center tour algorithm



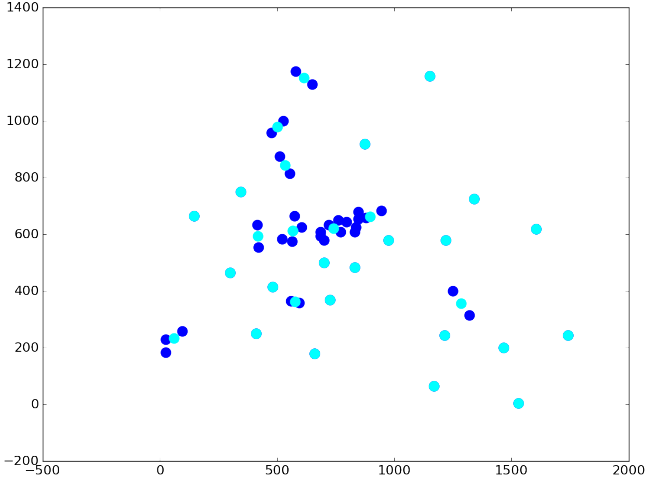
animation of the pair-center tour algorithm



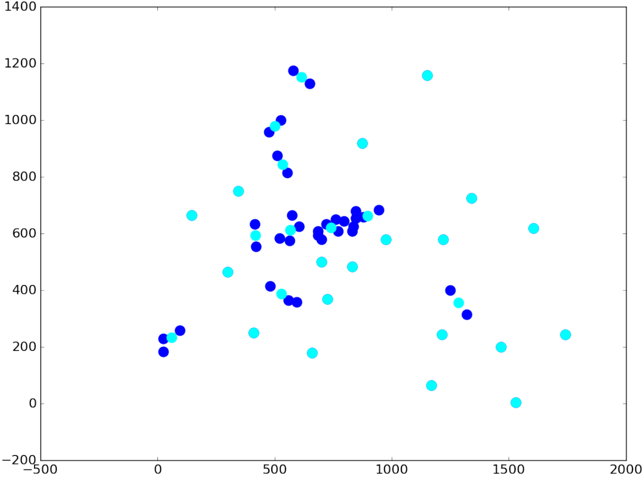
animation of the pair-center tour algorithm



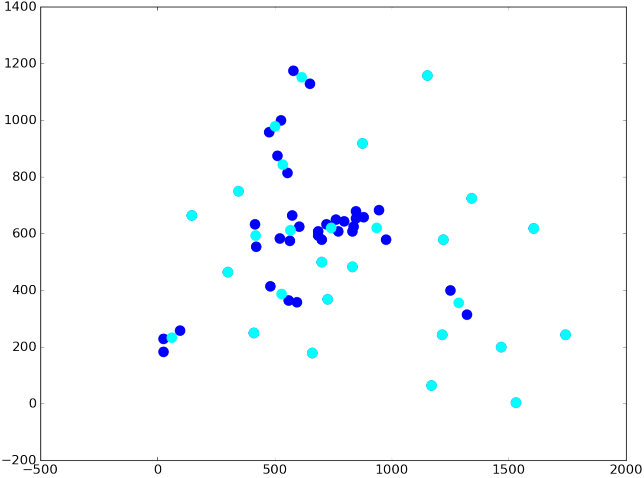
animation of the pair-center tour algorithm



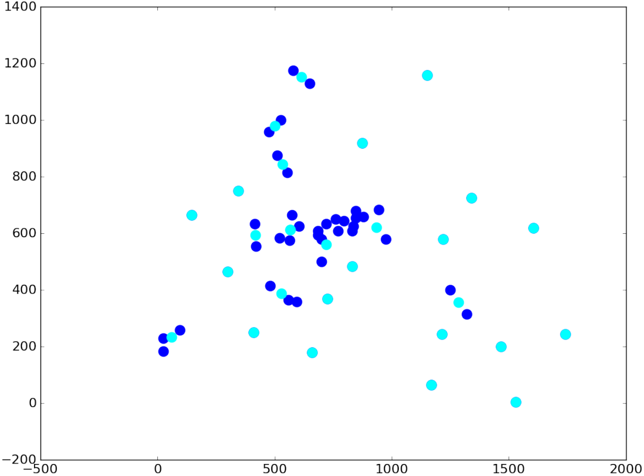
animation of the pair-center tour algorithm



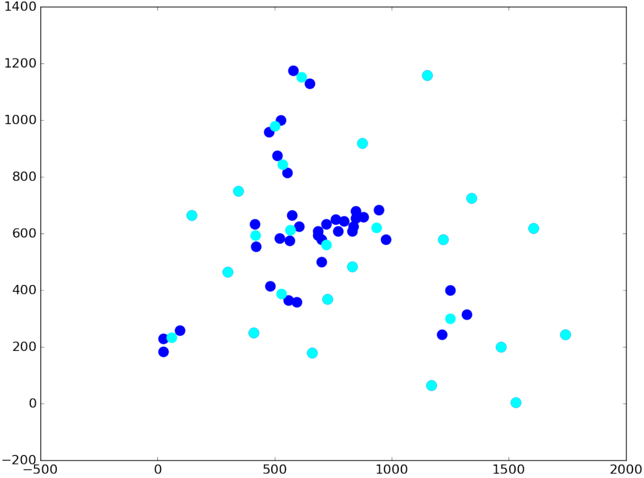
animation of the pair-center tour algorithm



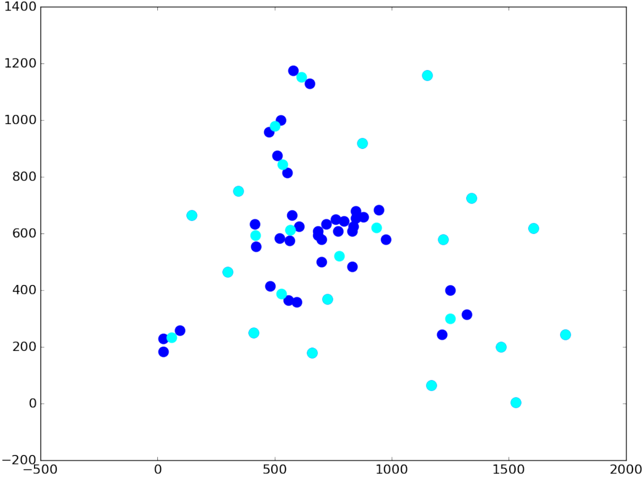
animation of the pair-center tour algorithm



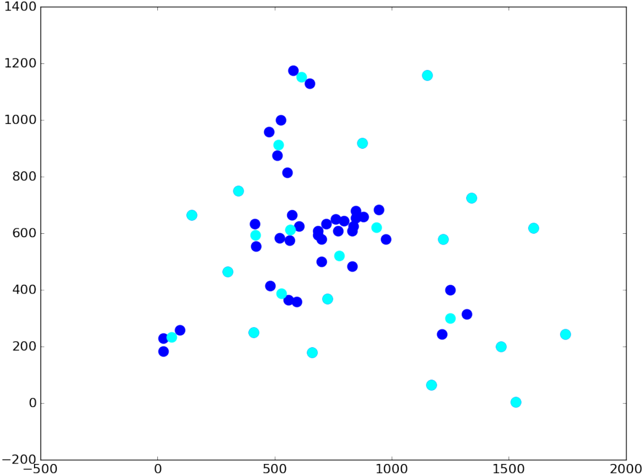
animation of the pair-center tour algorithm



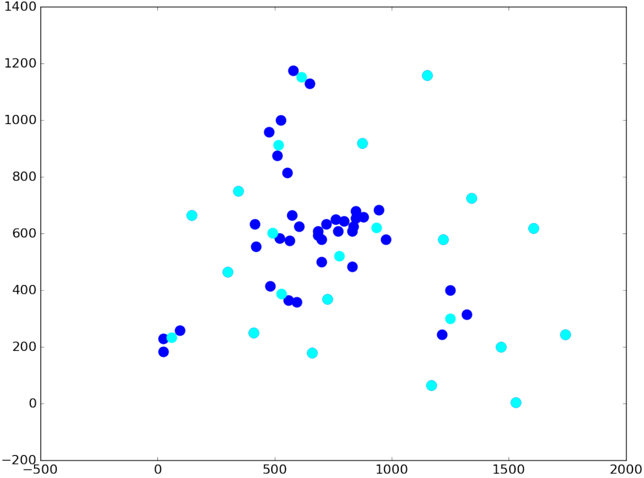
animation of the pair-center tour algorithm



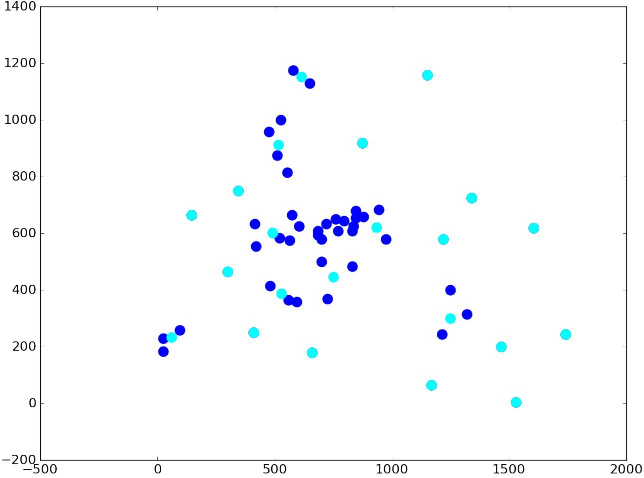
animation of the pair-center tour algorithm



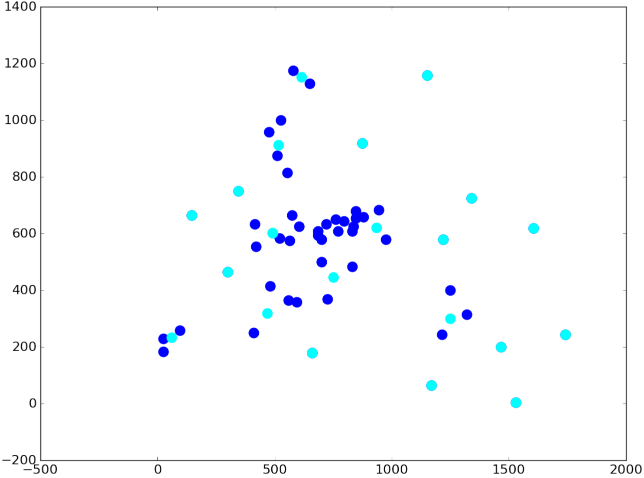
animation of the pair-center tour algorithm



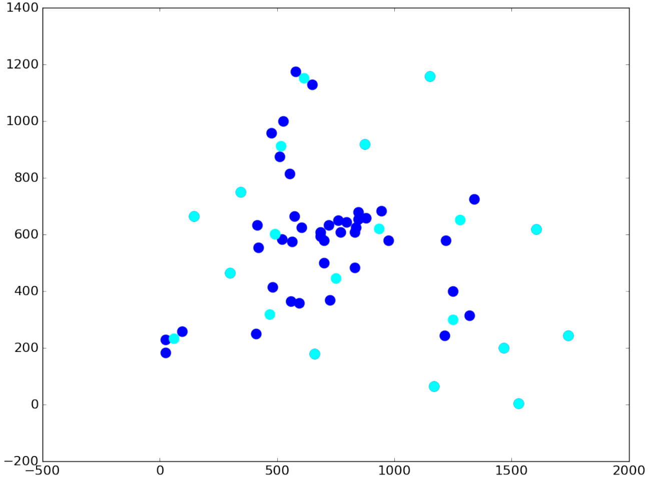
animation of the pair-center tour algorithm



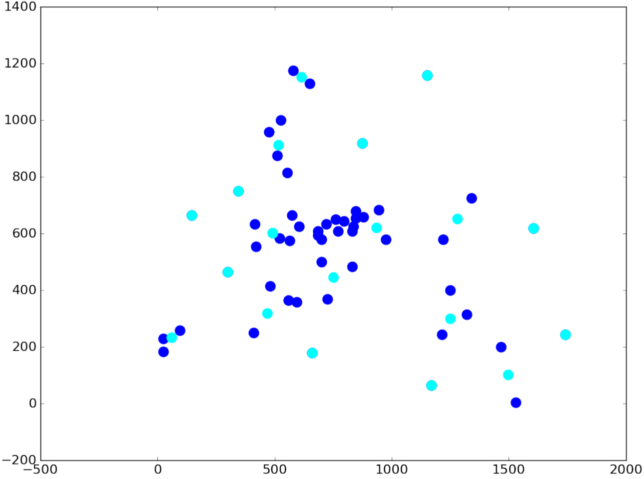
animation of the pair-center tour algorithm



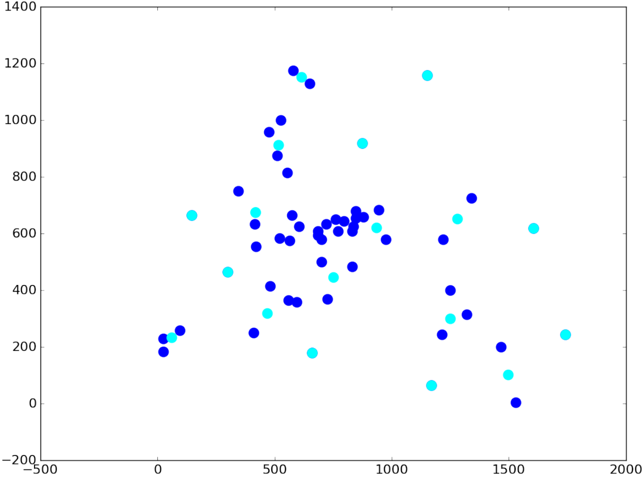
animation of the pair-center tour algorithm



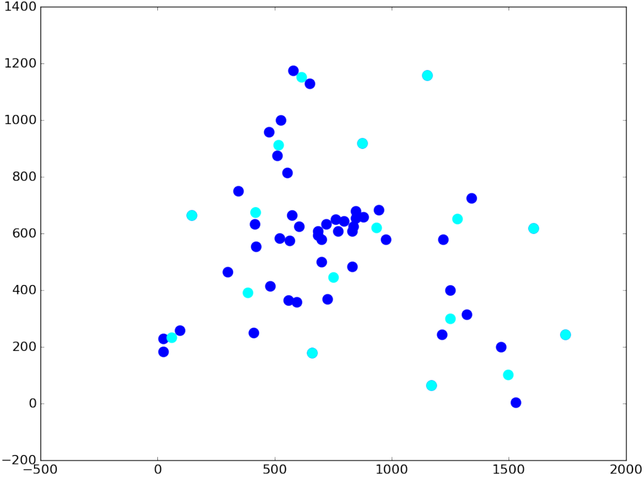
animation of the pair-center tour algorithm



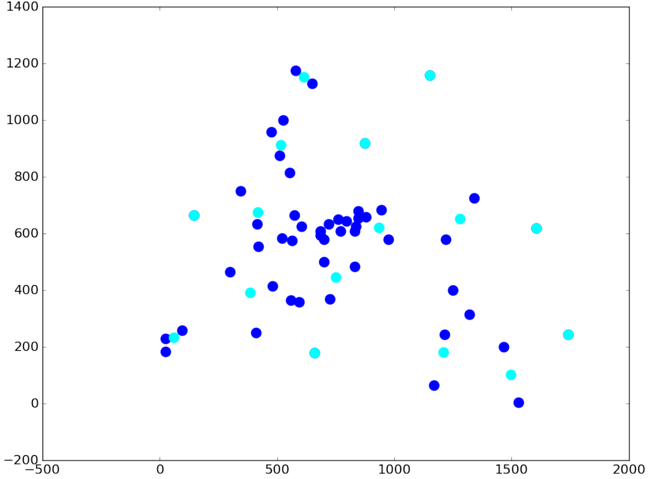
animation of the pair-center tour algorithm

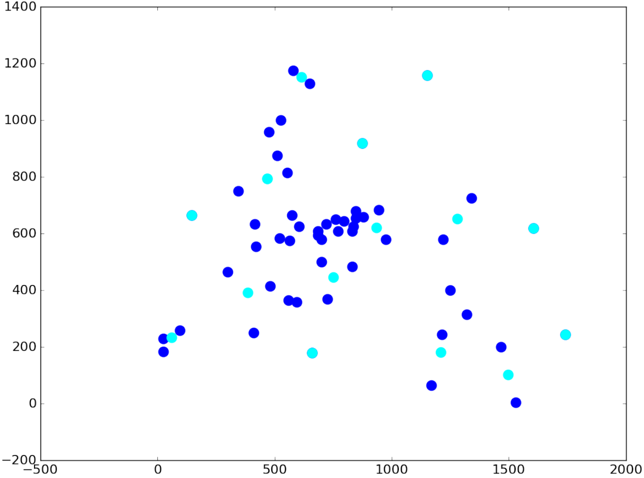


animation of the pair-center tour algorithm

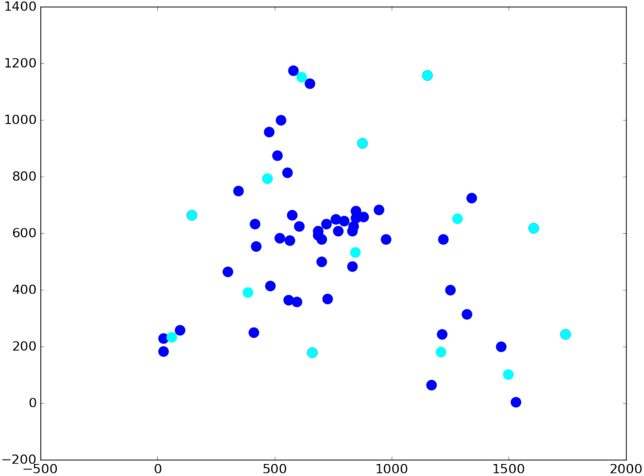


animation of the pair-center tour algorithm

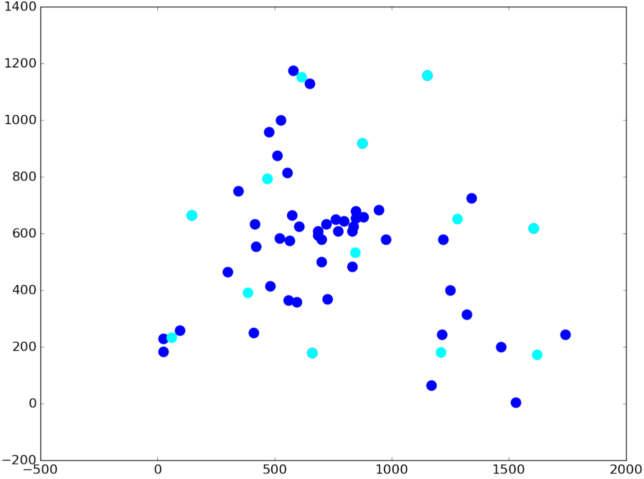



1400
1200
1000
800
600
400
200
0
−200
−500
0
500
1000
1500
2000

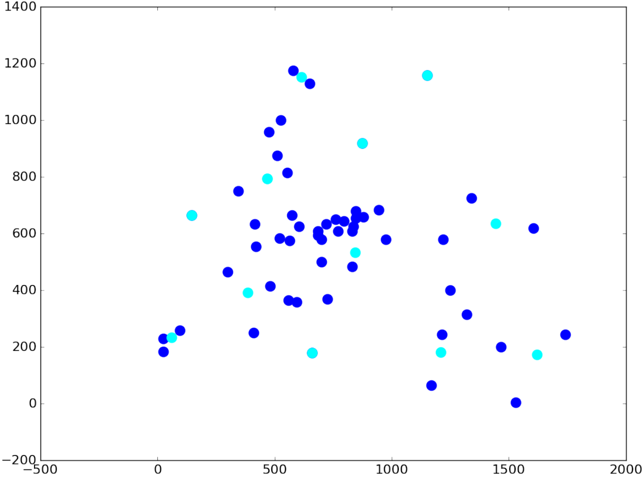
animation of the pair-center tour algorithm



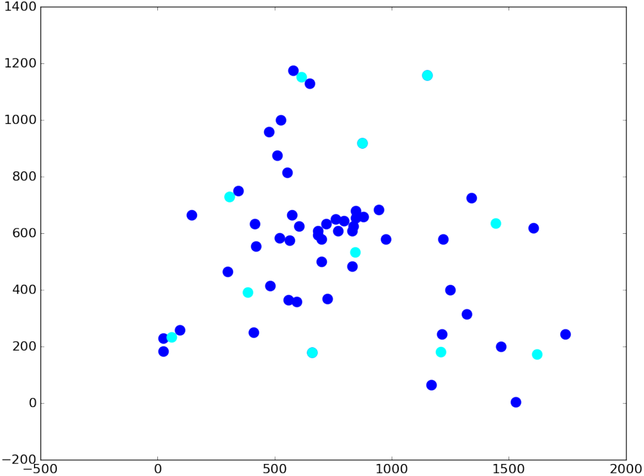
animation of the pair-center tour algorithm



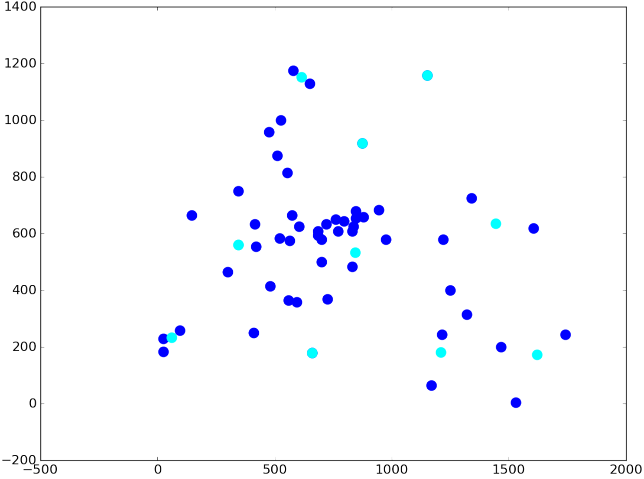
animation of the pair-center tour algorithm



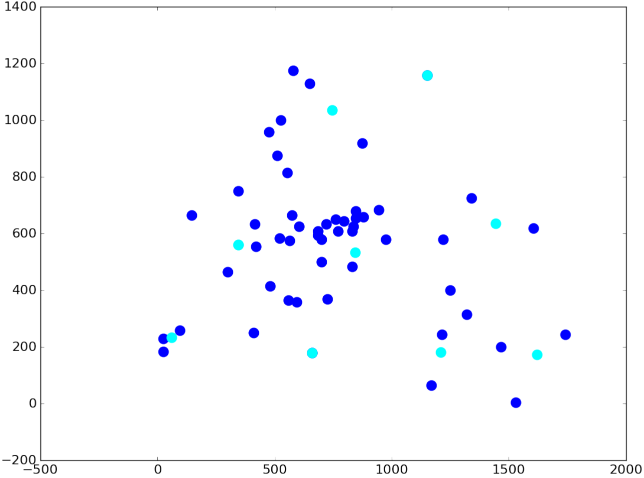
animation of the pair-center tour algorithm



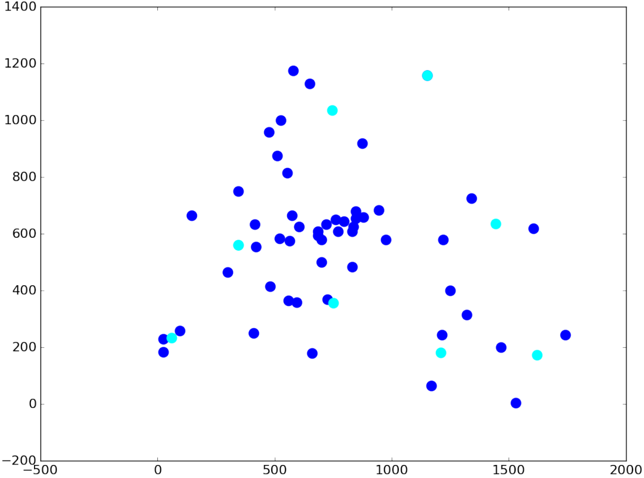
animation of the pair-center tour algorithm



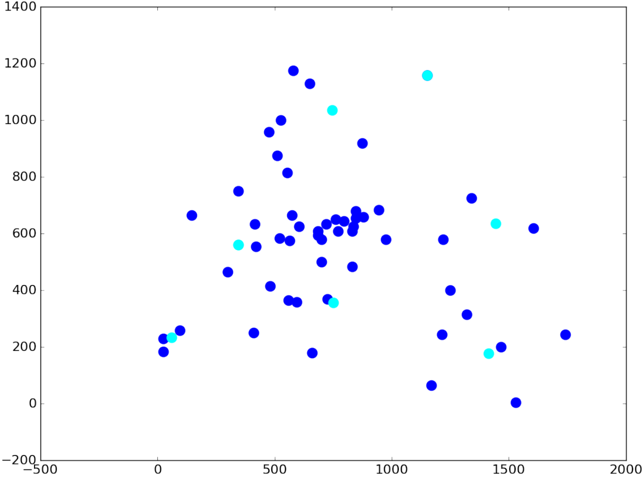
animation of the pair-center tour algorithm



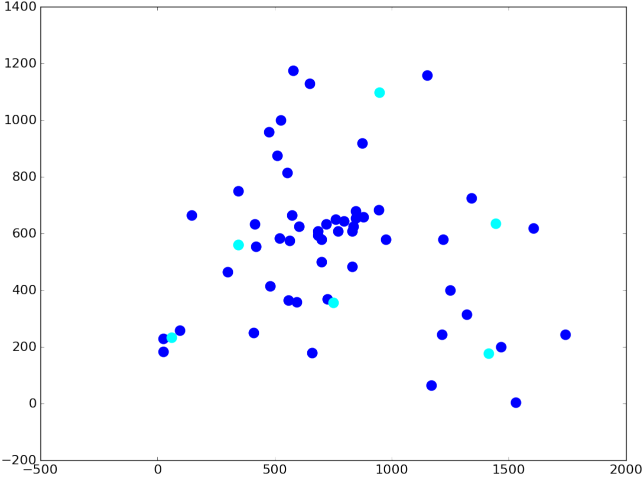
animation of the pair-center tour algorithm



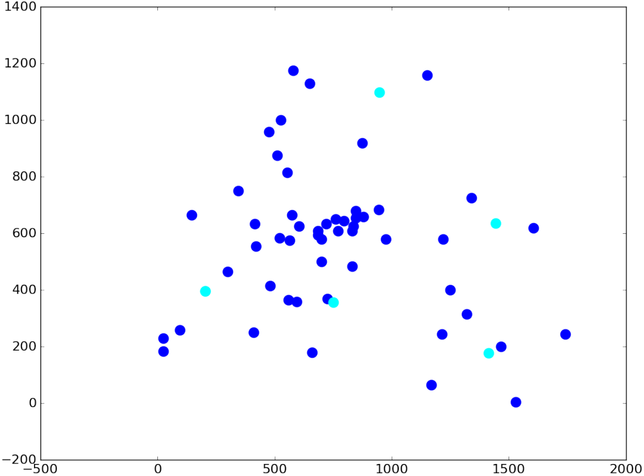
animation of the pair-center tour algorithm



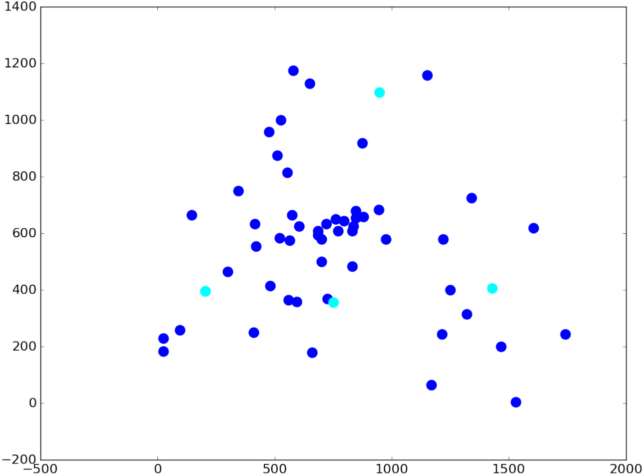
animation of the pair-center tour algorithm



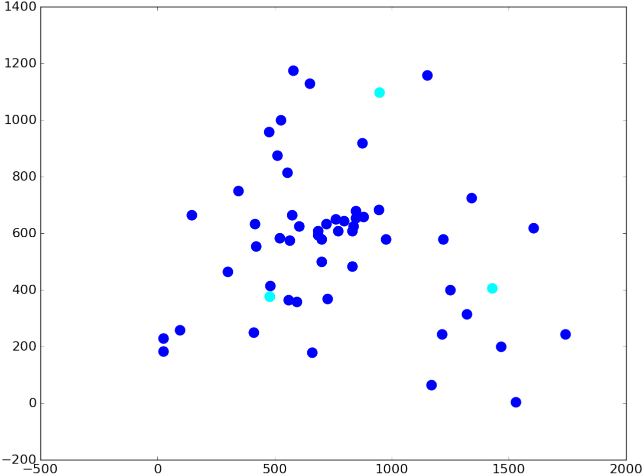
animation of the pair-center tour algorithm



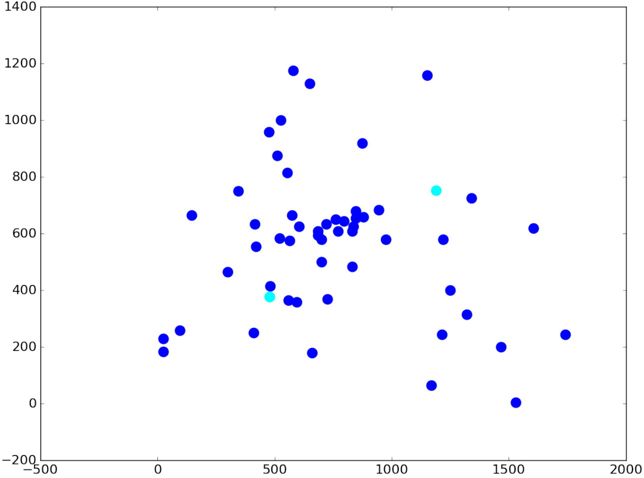
animation of the pair-center tour algorithm



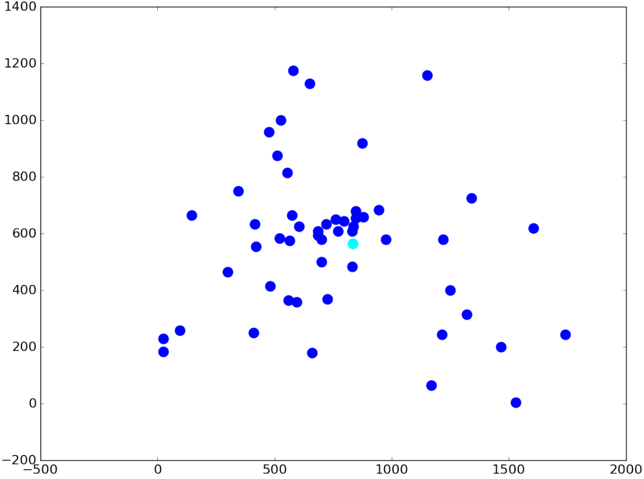
animation of the pair-center tour algorithm



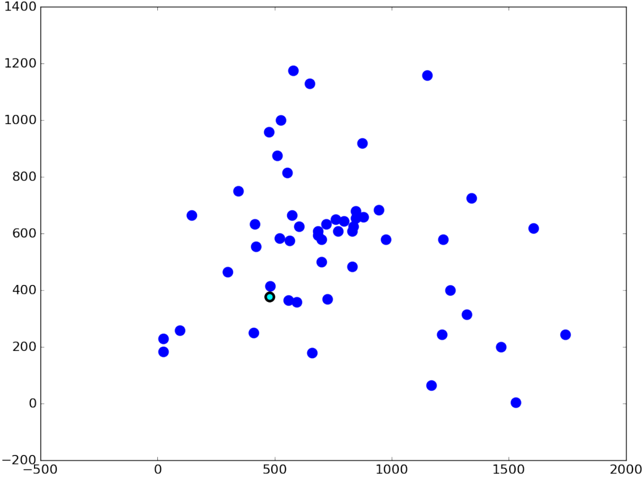
animation of the pair-center tour algorithm



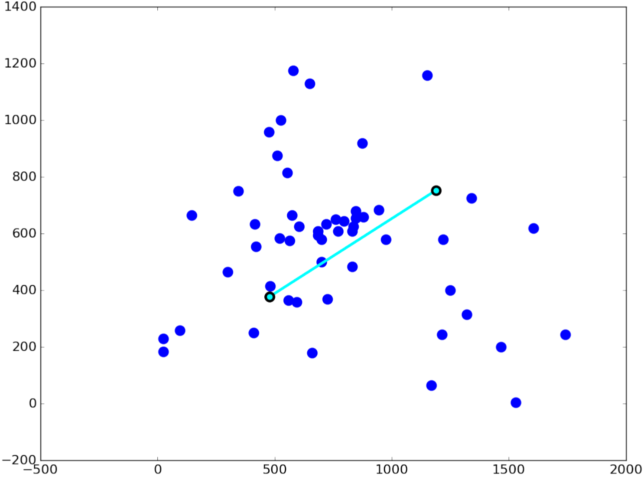
animation of the pair-center tour algorithm



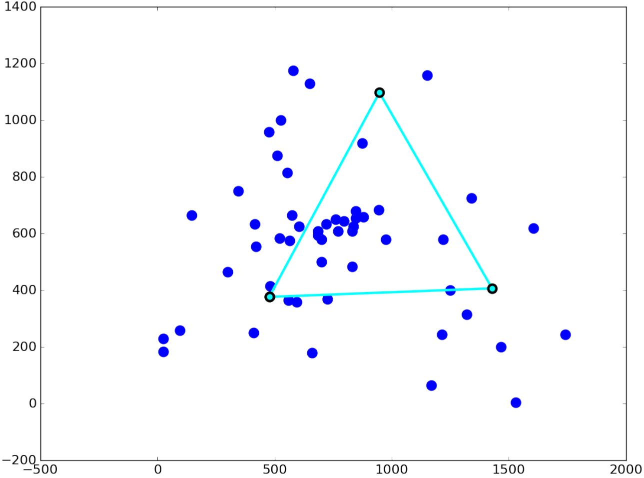
animation of the pair-center tour algorithm



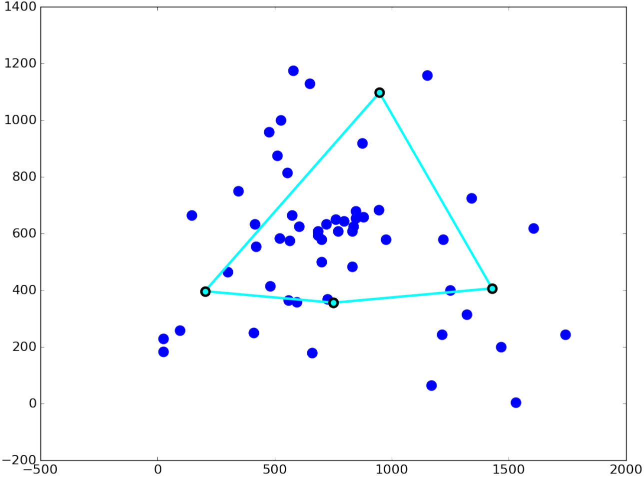
animation of the pair-center tour algorithm



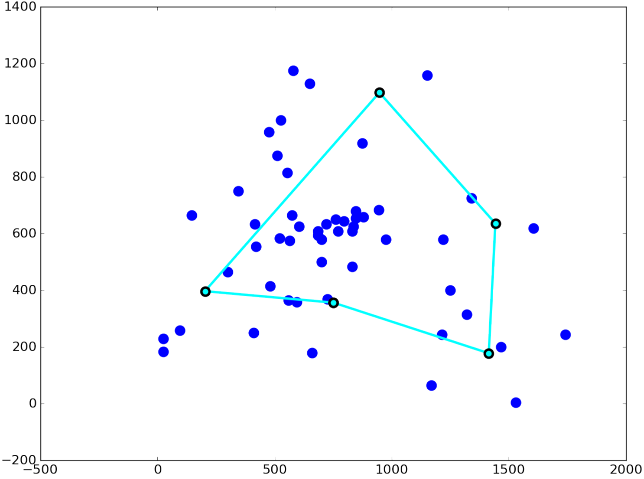
animation of the pair-center tour algorithm



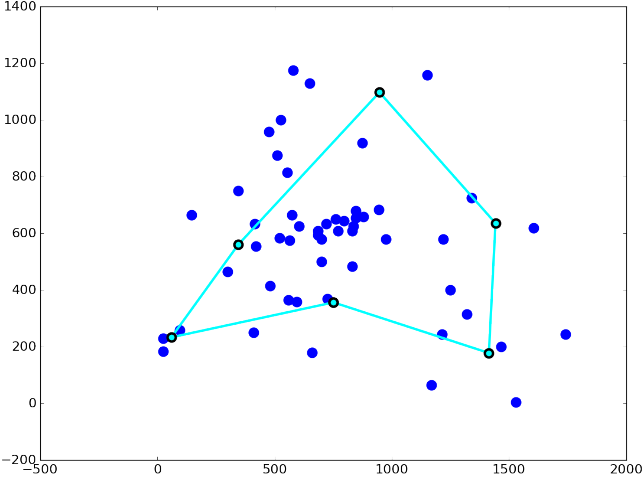
animation of the pair-center tour algorithm



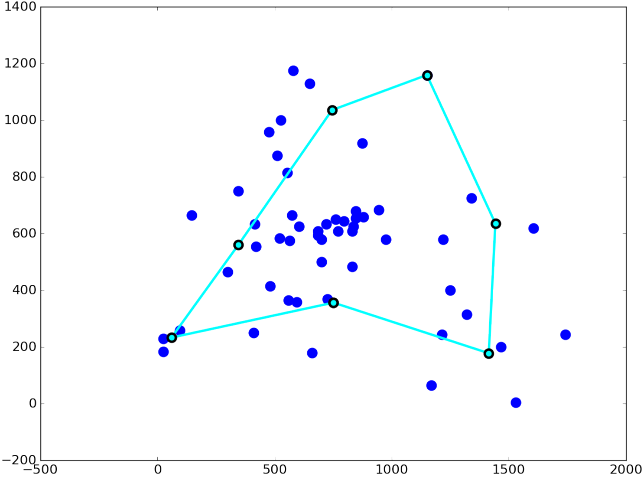
animation of the pair-center tour algorithm



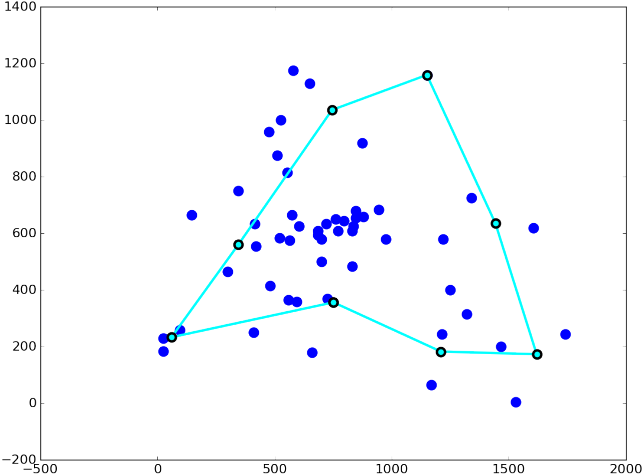
animation of the pair-center tour algorithm



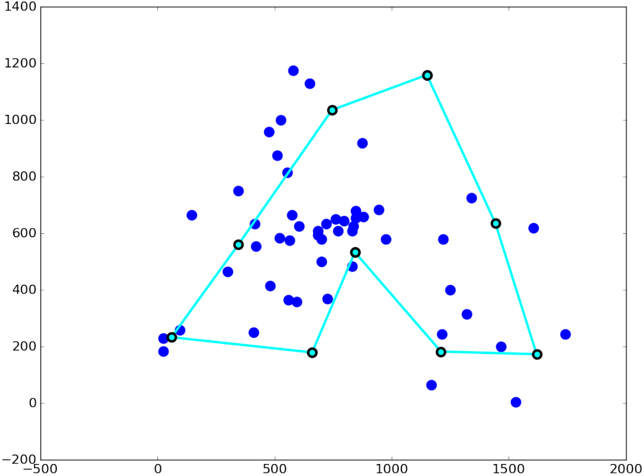
animation of the pair-center tour algorithm



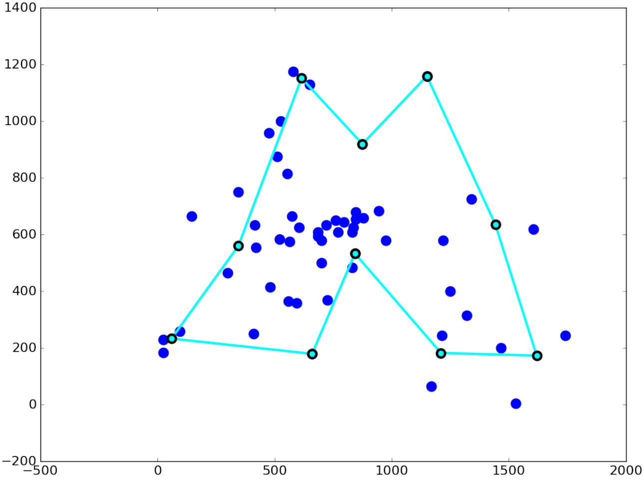
animation of the pair-center tour algorithm



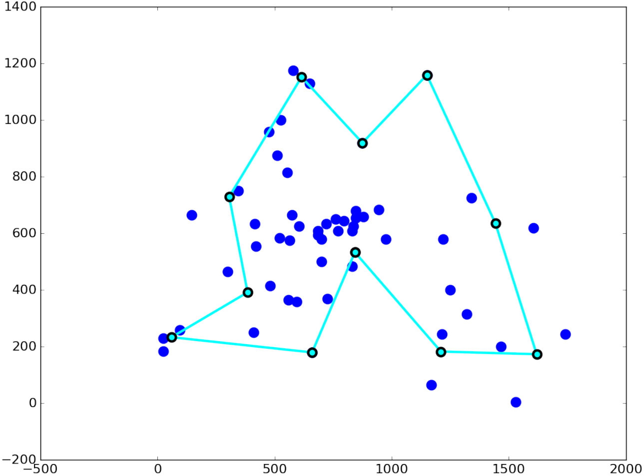
animation of the pair-center tour algorithm



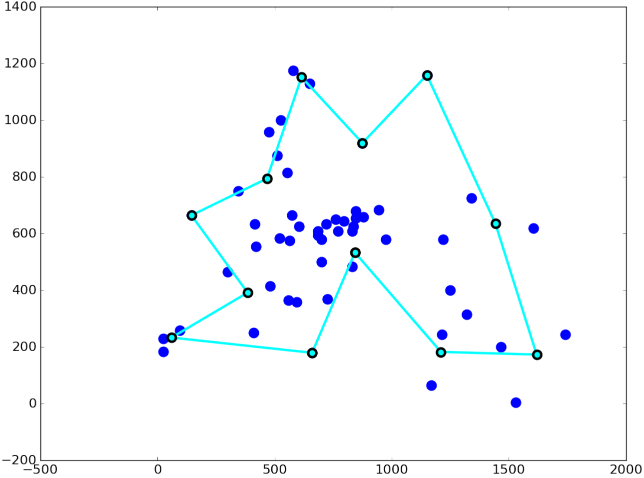
animation of the pair-center tour algorithm



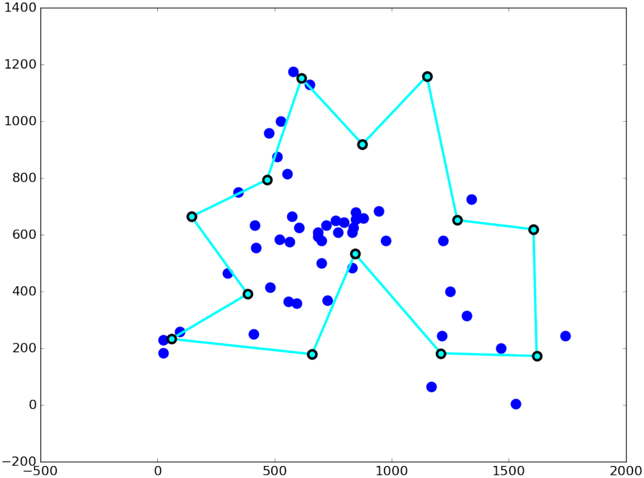
animation of the pair-center tour algorithm



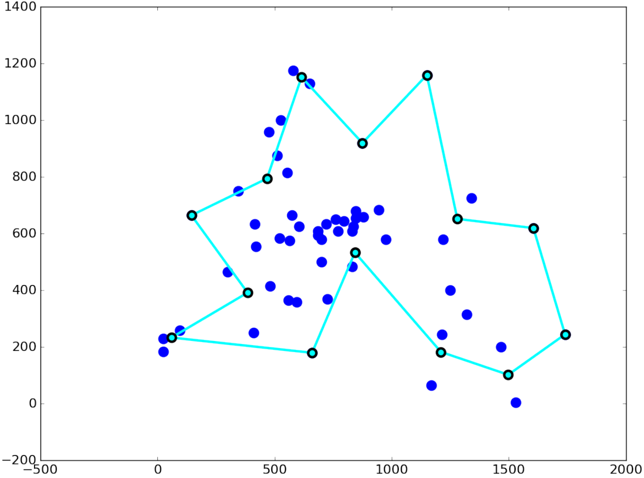
animation of the pair-center tour algorithm



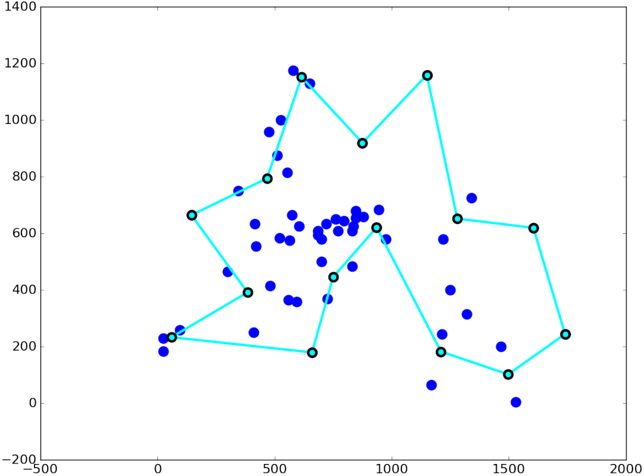
animation of the pair-center tour algorithm



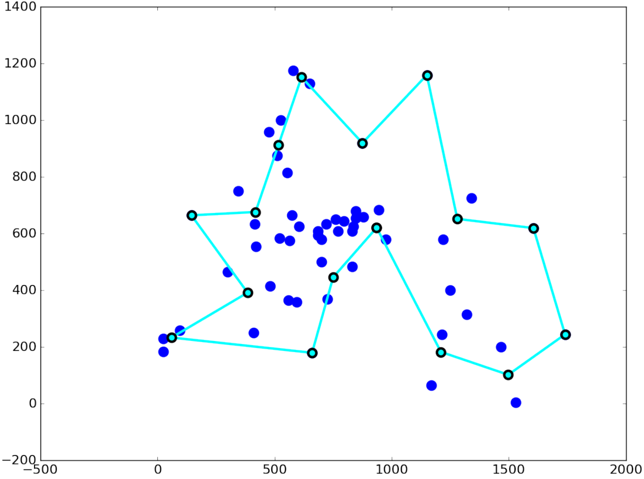
animation of the pair-center tour algorithm



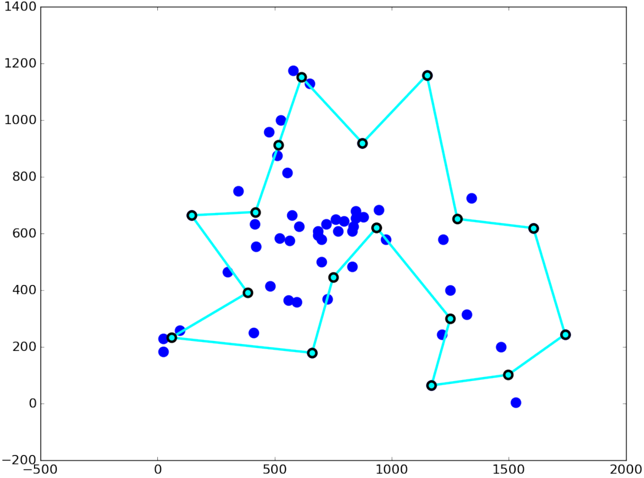
animation of the pair-center tour algorithm



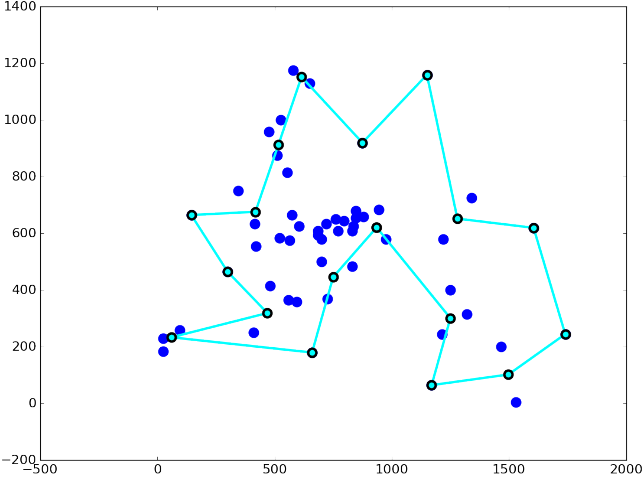
animation of the pair-center tour algorithm



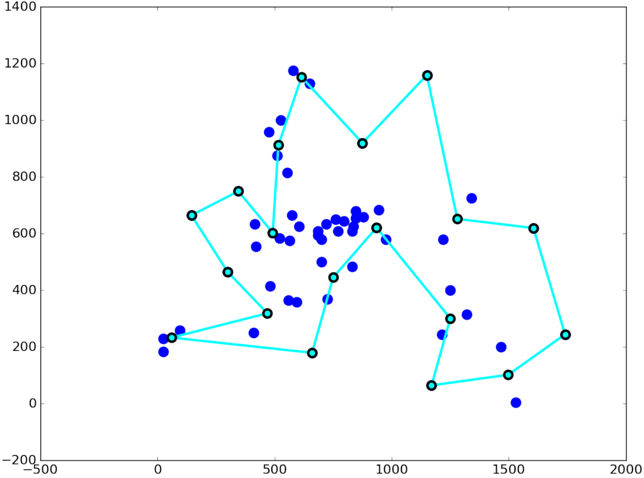
animation of the pair-center tour algorithm



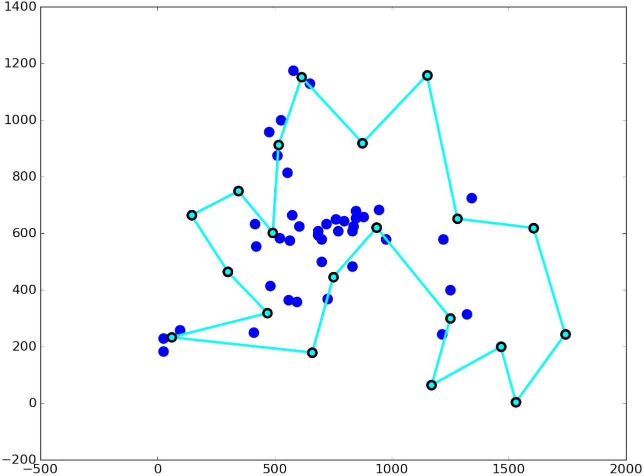
animation of the pair-center tour algorithm



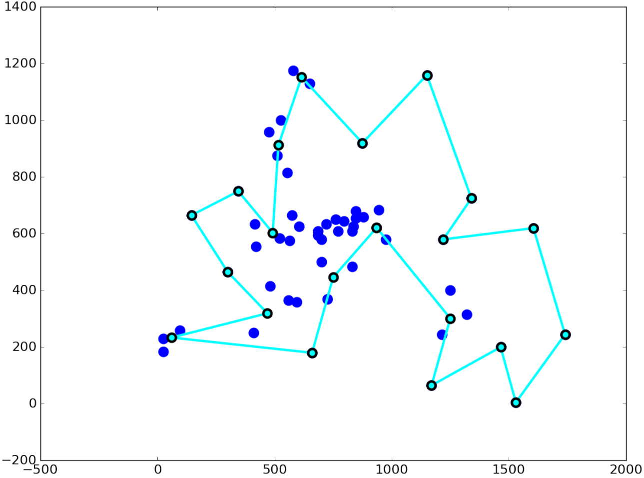
animation of the pair-center tour algorithm



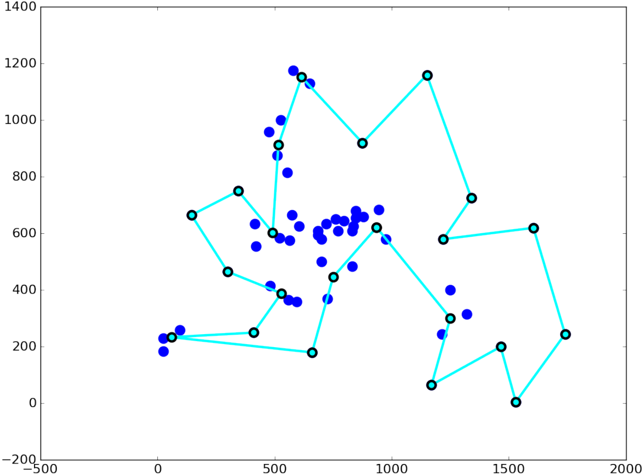
animation of the pair-center tour algorithm



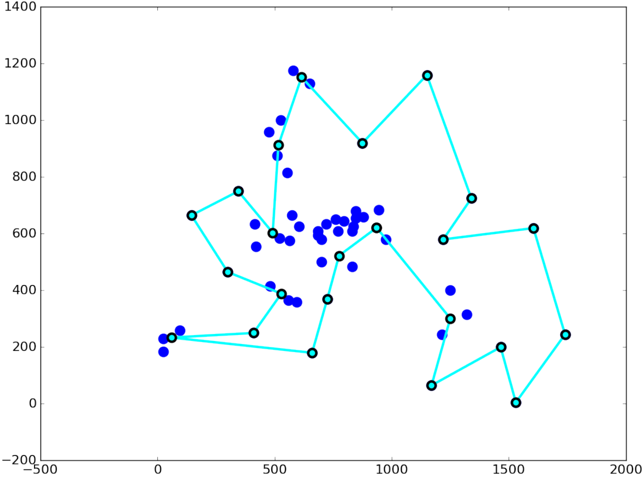
animation of the pair-center tour algorithm



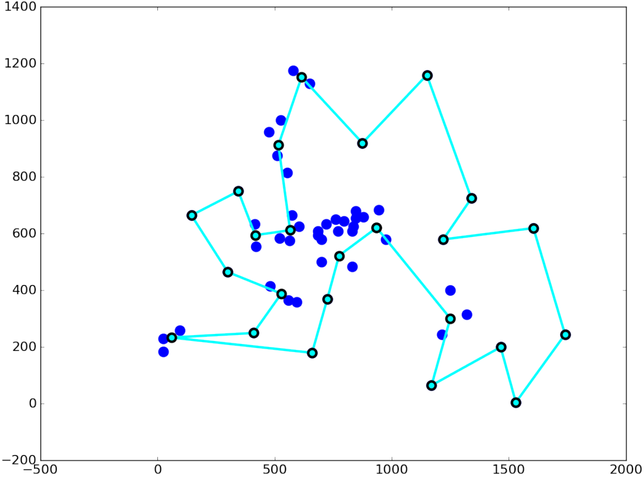
animation of the pair-center tour algorithm



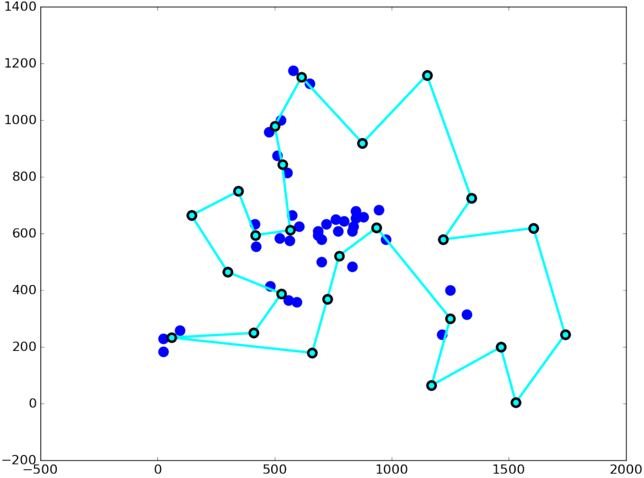
animation of the pair-center tour algorithm



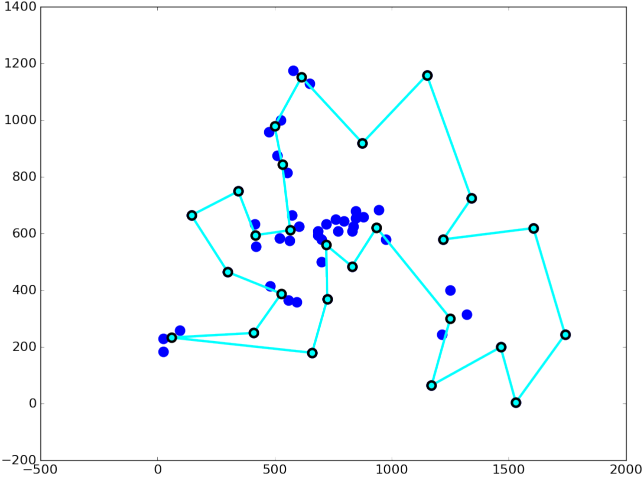
animation of the pair-center tour algorithm



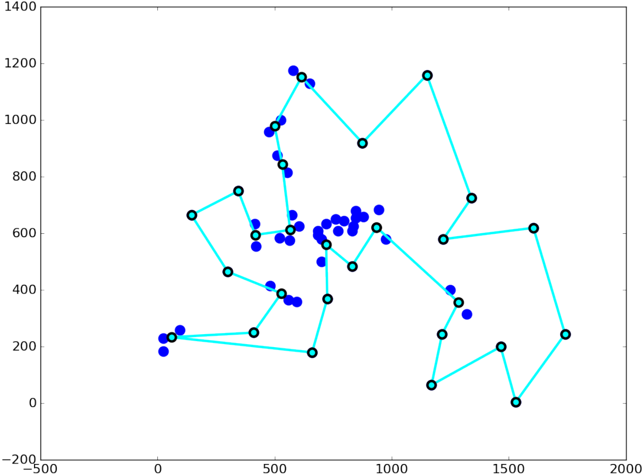
animation of the pair-center tour algorithm



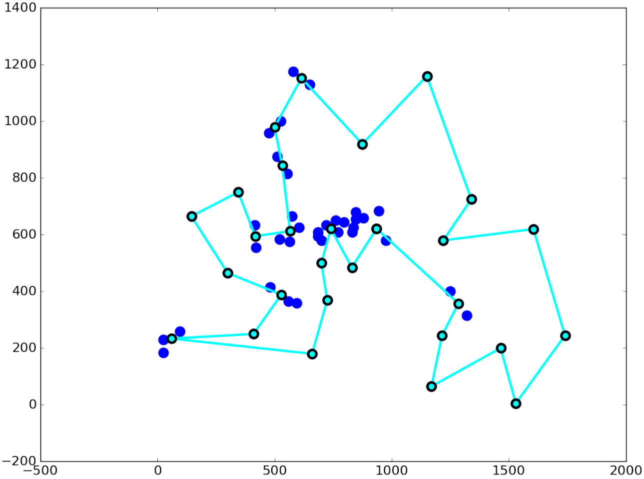
animation of the pair-center tour algorithm



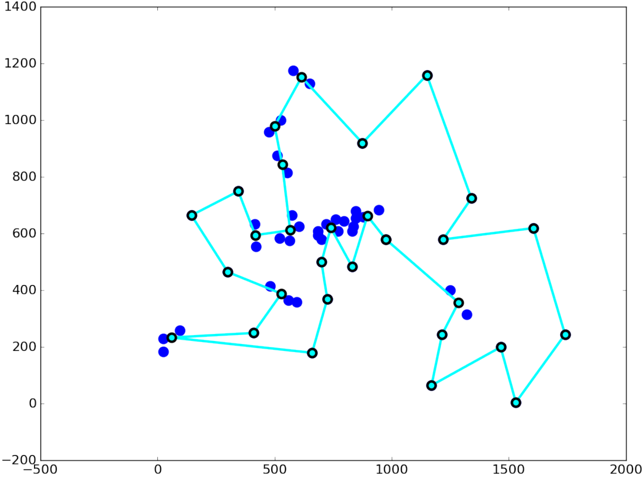
animation of the pair-center tour algorithm

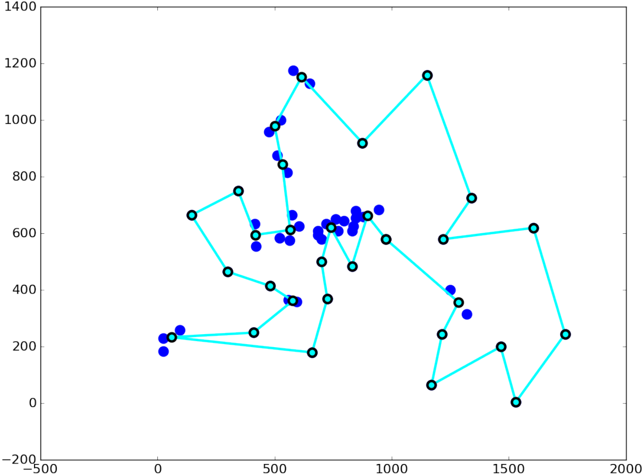


animation of the pair-center tour algorithm

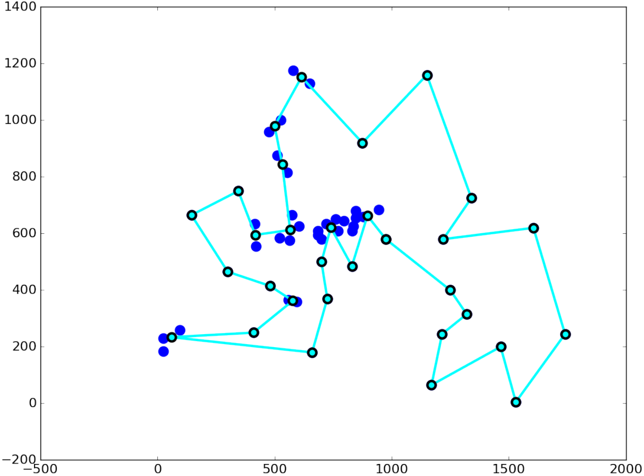


animation of the pair-center tour algorithm

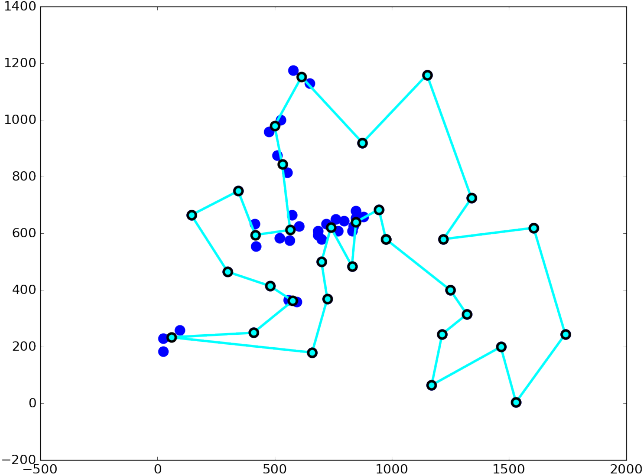



1400
1200
1000
800
600
400
200
0
−200
−500
0
500
1000
1500
2000

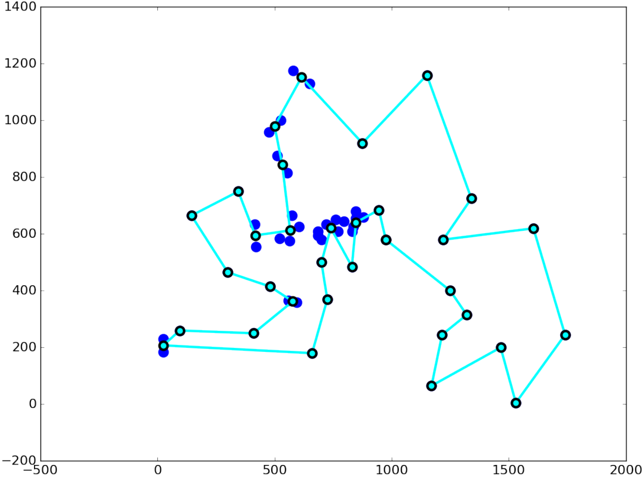
animation of the pair-center tour algorithm



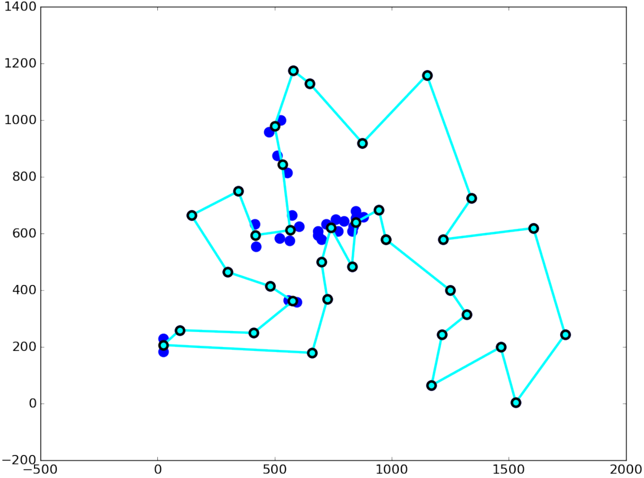
animation of the pair-center tour algorithm



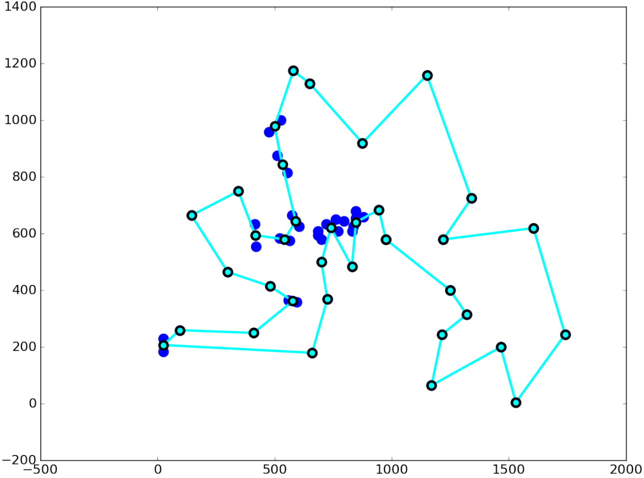
animation of the pair-center tour algorithm



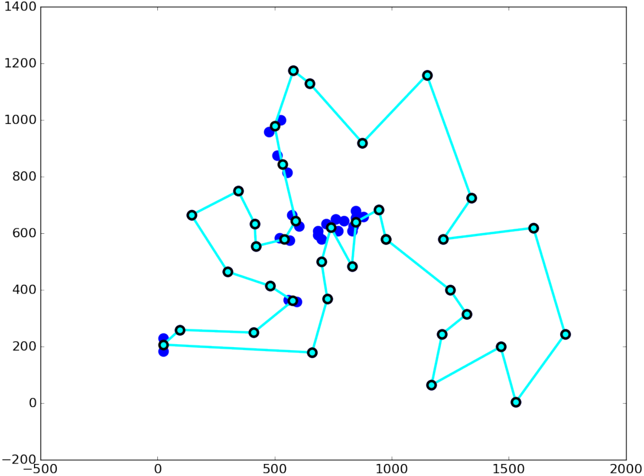
animation of the pair-center tour algorithm



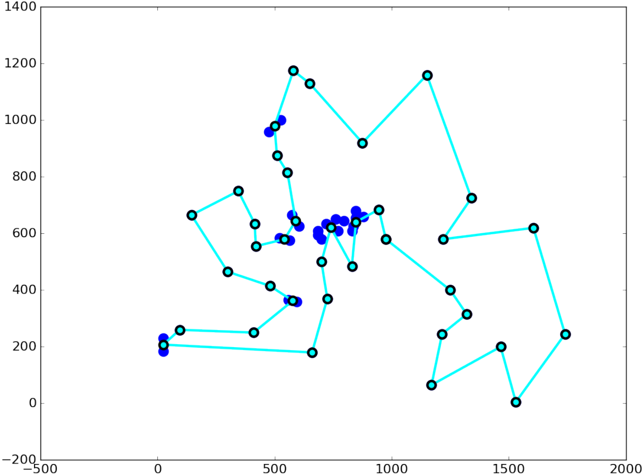
animation of the pair-center tour algorithm



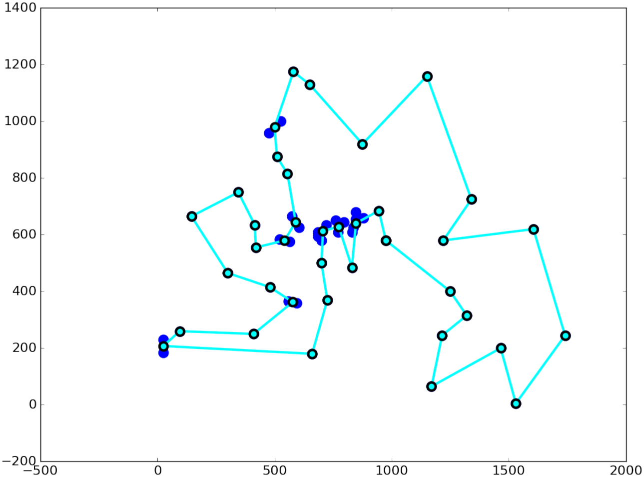
animation of the pair-center tour algorithm



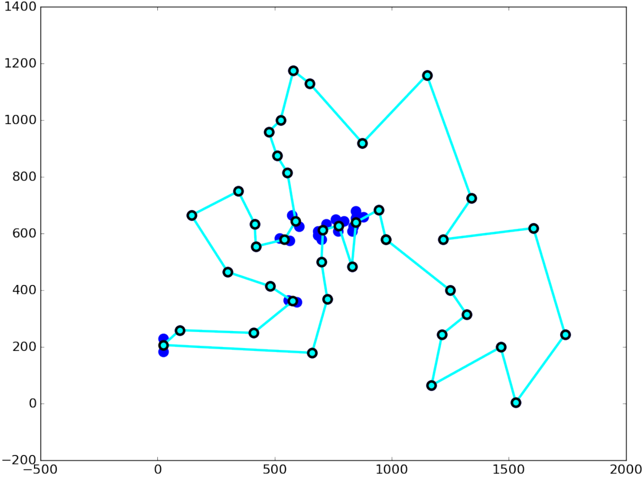
animation of the pair-center tour algorithm



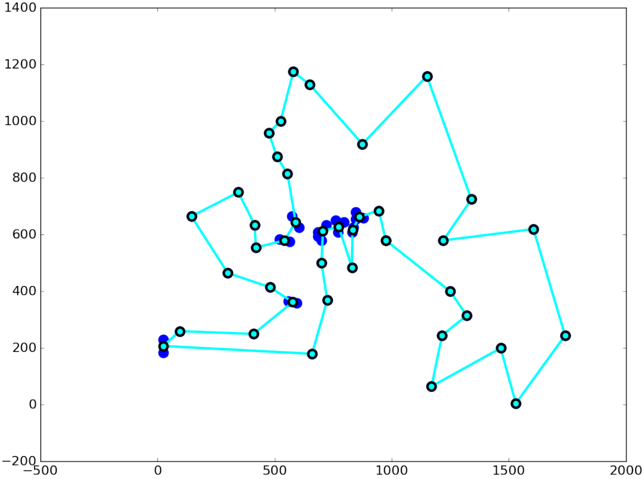
animation of the pair-center tour algorithm



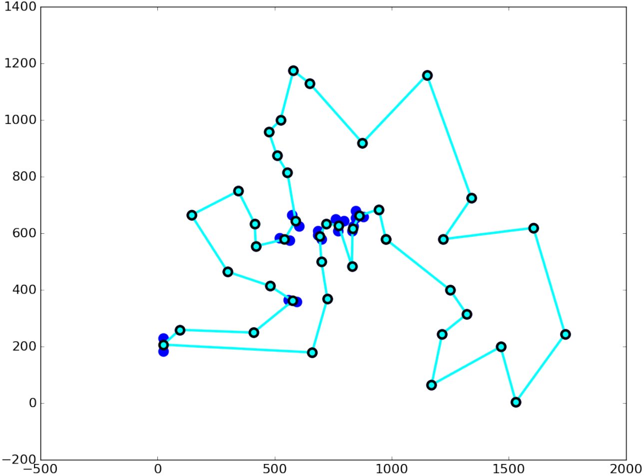
animation of the pair-center tour algorithm



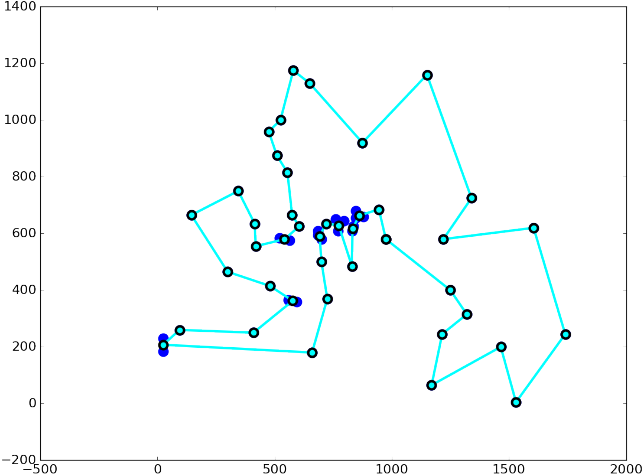
animation of the pair-center tour algorithm



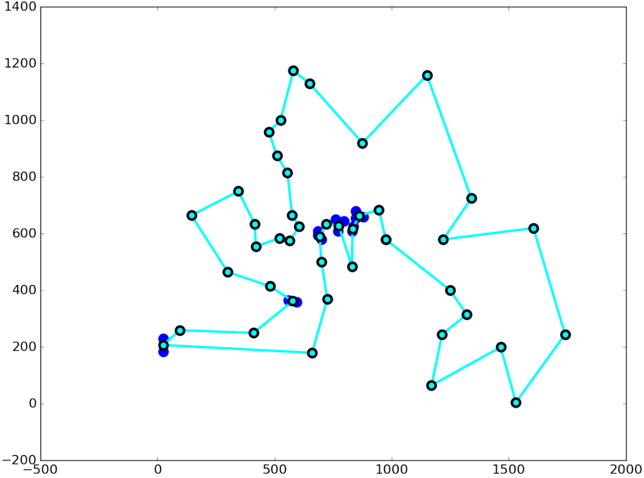
animation of the pair-center tour algorithm



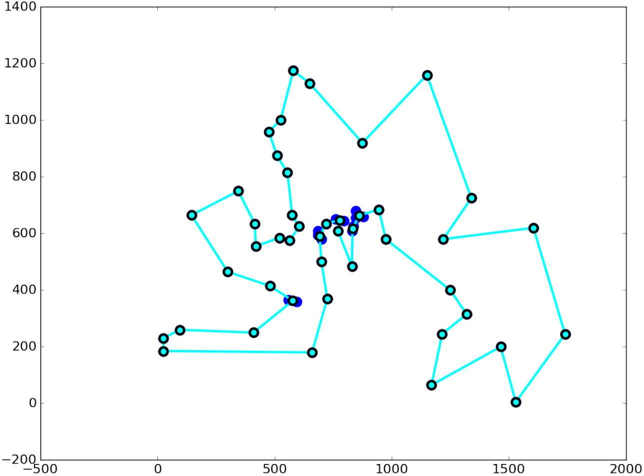
animation of the pair-center tour algorithm



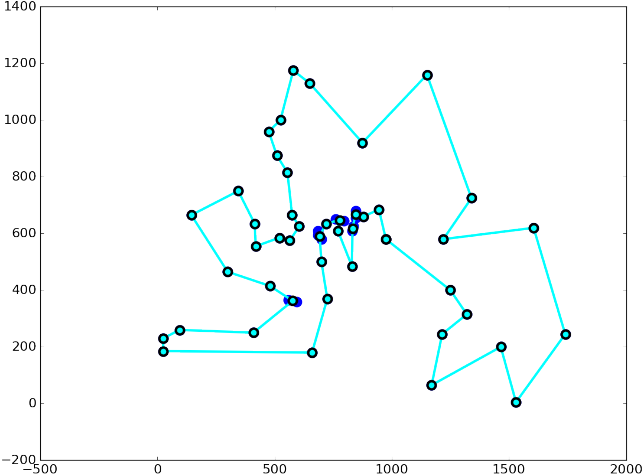
animation of the pair-center tour algorithm



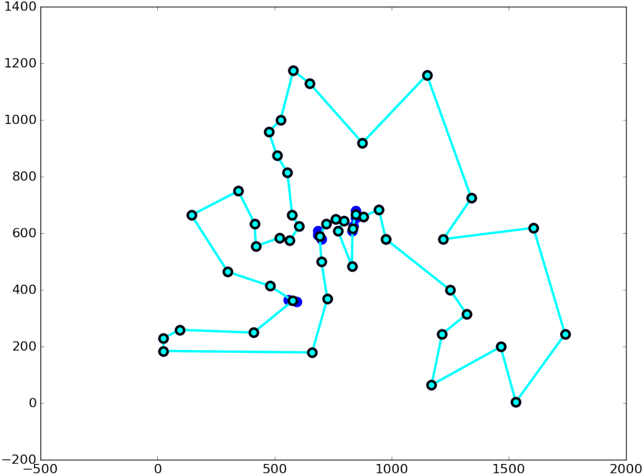
animation of the pair-center tour algorithm



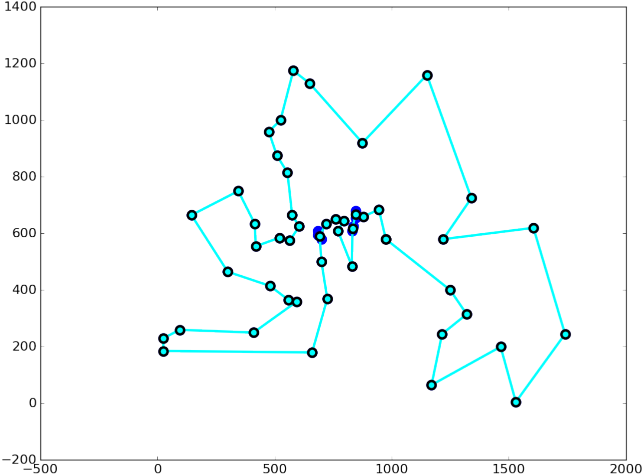
animation of the pair-center tour algorithm



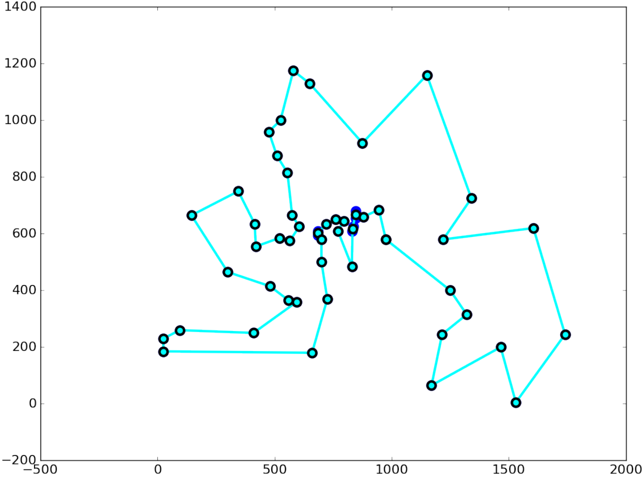
animation of the pair-center tour algorithm



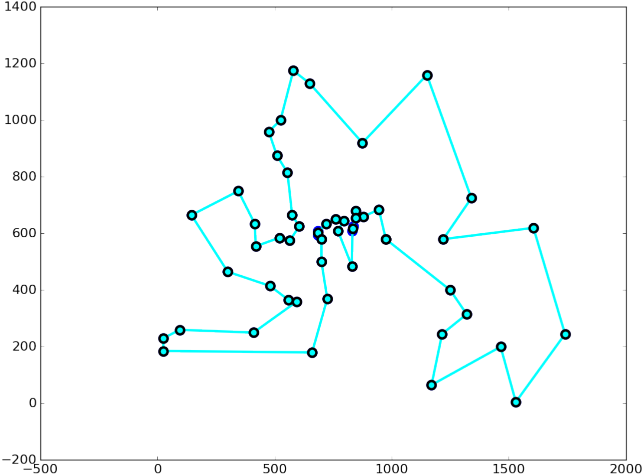
animation of the pair-center tour algorithm



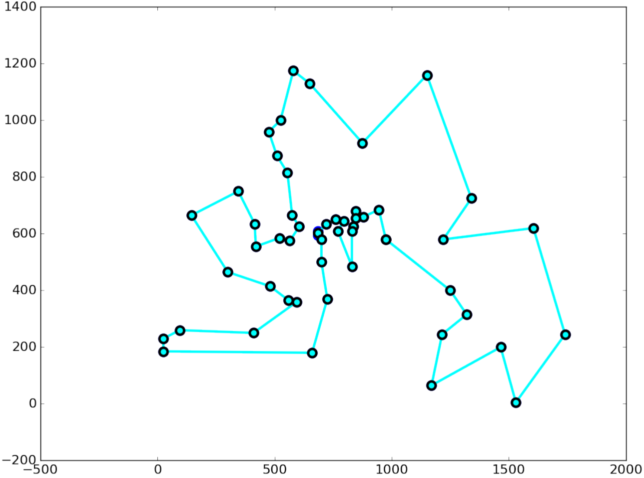
animation of the pair-center tour algorithm



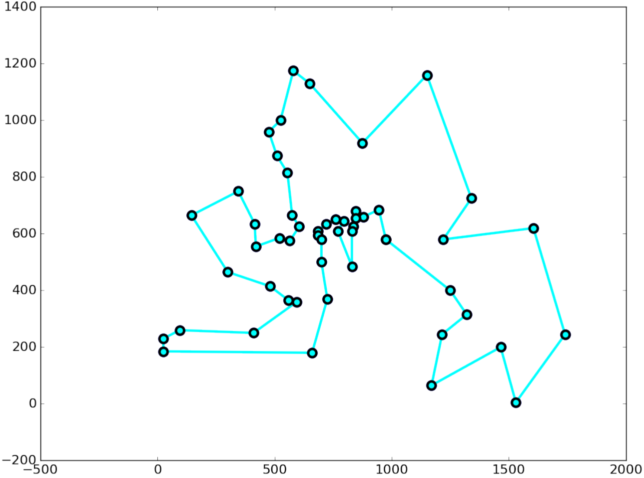
animation of the pair-center tour algorithm



animation of the pair-center tour algorithm



animation of the pair-center tour algorithm



the TSP length-line sum-up

length of tour

L_{CN}	closest-neighbor tour (can be large, 1974)
$2 \cdot L_{opt}$	minimum spanning tree algorithm (or quick tour)
$1.5 \cdot L_{opt}$	Christofides algorithm (best known, 1976)
L_{play}	playground for good heuristic algorithms
L_{opt}	optimal tour length, Bellman-Held-Karp algorithm (1962)
L_{HKB}	*Held-Karp bound* $((L_{opt} - L_{HKB})/L_{opt} \approx 0.01$*, 1971)*
L_{MDB}	*minimal distance bound (possible* $L_{MDB} = L_{opt}$*)*
0	absolute minimum

How does the genetic algorithm work?

A genetic algorithm is a bio-inspired probabilistic algorithm:

- initialize a set of individuals
- while *stopping criterium* not met
  - evaluate fitness of the individuals (in search space)
  - generate off-springs (mutation and crossover, in the encoding space)
  - generate a new generation, i.e., a subset of parents plus off-springs (selection)
- report best individual generated in the process

You'll work with this approach in the lab hours.

additional information

additional information and benchmark instances can be found at:

- http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/ or
- https://www.math.uwaterloo.ca/tsp/data/index.html
- almost a counterexample of how to implement GA for TSP https://jaketae.github.io/study/genetic-algorithm/
- we use the work at https://github.com/guofei9987/scikit-opt

parallelization

- Monte Carlo algorithms, and hence, evolutionary algorithms are often quite easy to parallelize.
- We will not talk about parallelization in this course, however, it's an important issue in order to achieve good performance on modern systems.