Evolutionary Computation

2024/25

Master Artificial Intelligence

Arno Formella

Departamento de Informática
Escola Superior de Enxeñaría Informática
Universidade de Vigo

24/25

randomized or probabilistic algorithms

two main classes:

- Monte Carlo algorithm:
  for a certain number of iterations do:
  - perform some randomized algorithm step towards a better solution
  - (usually keep track of your best solution found so far)
- Monte Carlo algorithms always terminate and (hopefully) find a somewhat good solution.
- Las Vegas algorithm:
  while a certain end condition still is not met do:
  - perform some randomized algorithm step towards a better solution
- Las Vegas algorithms only terminate with a correct solution (or do not terminate at all), but their runtime is probabilistic.

classical algorithmic paradigms

- backtracking
- branch and bound
- brute-force (or exhaustive) search
- divide and conquer
- dynamic programming
- greedy algorithm
- prune and search
- online algorithms

What are heuristic algorithms?

- Just do something (you come up with)
- and be happy (with the properties of the result).

What are evolutionary algorithms?

- Evolutionary algorithms are heuristic optimization algorithms usually implemented with the Monte Carlo approach (and possibly a Las Vegas stopping condition when available)
- that exhibit, let's say, at least a tendency to approach a global minimum as solution of the optimization problem.
- Often they are inspired by some phenomenon observable in nature (or a *creative* name has been used).
- for evolutionary algorithms often much weaker properties are accepted for the output configuration, i.e., for TSP the guaranteed property is just to have at least a tour, but the tour might be arbitary far from the optimum

optimization

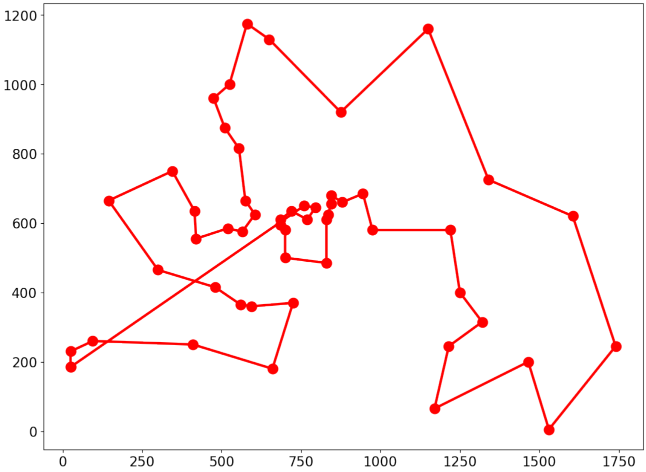
- Given a search space $\mathbb{X}$ (called as well *search domain* or *problem space*) and
- a function f (bounded from below) from the search space to the real numbers (or at least a totally ordered set), e.g. $f : \mathbb{X} \longrightarrow \mathbb{R}$,
- find an element $x^\star \in \mathbb{X}$ such that $f(x^\star) \leq f(x)$ for all $x \in \mathbb{X}$.
- i.e., we look for a global minimum.

Observe: whenever we look for a maximum, we can use just a negative sign and look for a minimum (and f must be bounded from above)!

local versus global minimum

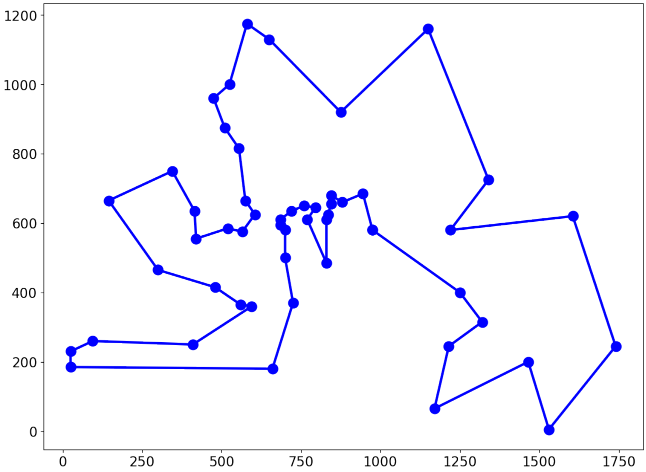
- If we can determine a neighborhood around each element $x \in \mathbb{X}$, we call $\mathcal{N}(x)$ the set of neighbors of x.
- and if we have for all such neighbors $x' \in \mathcal{N}(x)$, that $f(x) \leq f(x')$,
- then we call x a local minimum (sometimes written as $\hat{x}$).
- Reaching a local minimum is often somewhat easier, as we can take advantage of a possibly available gradient (local search algorithms).
- It happens to be an issue in optimization not to get stuck in a local minimum while searching for a global minimum, which is the ultimate goal.
- Often relative error or *gap* is used to qualify the solution: $100 \cdot (L - L_{opt})/L_{opt}$

traveling salesperson problem: first impressions



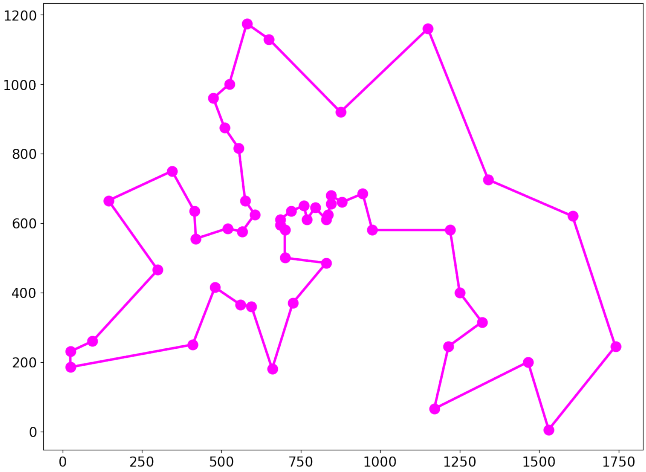
a closest-neighbor tour: 8.49% gap

traveling salesperson problem: first impressions



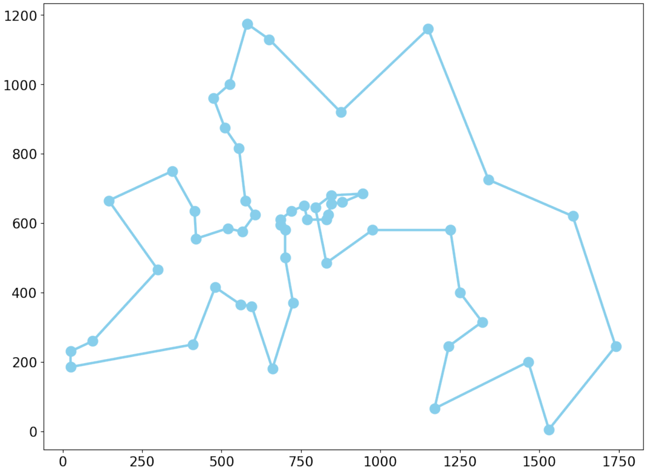
a pair-center tour: 7.28% gap

traveling salesperson problem: first impressions



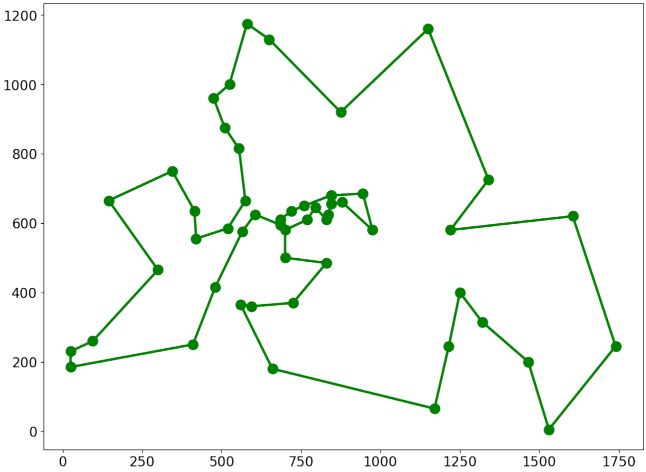
the best tour (known for this example): 0% gap

traveling salesperson problem: first impressions



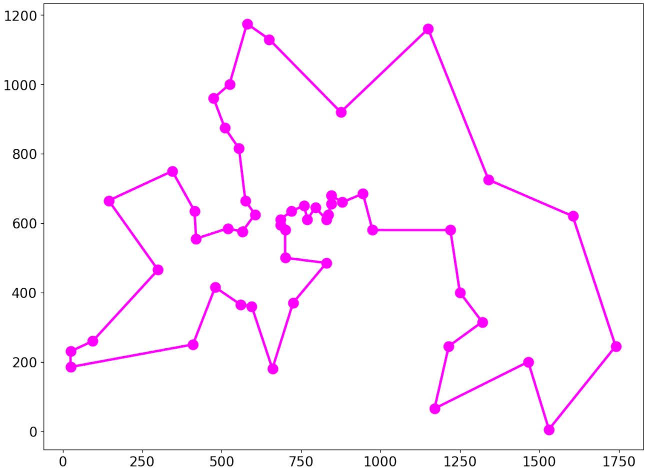
a quick tour (with Monte Carlo): 2.32% gap

traveling salesperson problem: first impressions



a genetic algorithm tour: 8.37% gap

traveling salesperson problem: first impressions



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

evaluation of approximative algorithm

Please ask yourself the question: How would you evaluate approximation algorithms aimed to provide practical solutions to complex problems, especially in order to compare different approaches? For exact algorithms this is easy:

- first prove the algorithm is correct and correcly implemented
- analyse its runtime (according to input size)
- analyse its memory requirements

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
  - computing the bound is an iterative process as well
    (but in polynomial time, or linear time for randomized
    approximation)
  - can be as worse as 2/3 times optimal length

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 $\leq 3/2 \cdot L_{opt}$
  runtime $\mathcal{O}(n^3)$

known solving algorithms for TSP

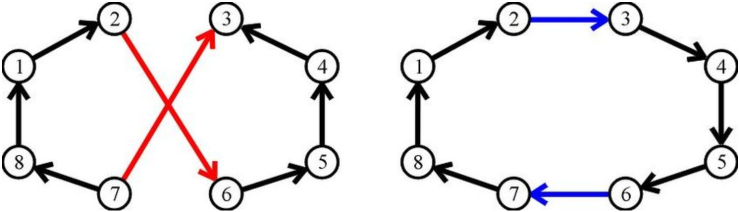
- brute force exhaustive search $\mathcal{O}(n!)$
  (quite easy to implement)
- Bellman-Held-Karp dynamic programming for Euclidean TSP $\mathcal{O}(n^2 2^n)$ time and $\mathcal{O}(n2^n)$ space
  (not covered here, please refer to advanced algorithms in computer science)
- state-of-the-art solver Concorde
  https://www.math.uwaterloo.ca/tsp/concorde.html
- state-of-the-art approximative solver LKH
  http://webhotel4.ruc.dk/~keld/research/LKH-3/

known good approximation heuristics for TSP

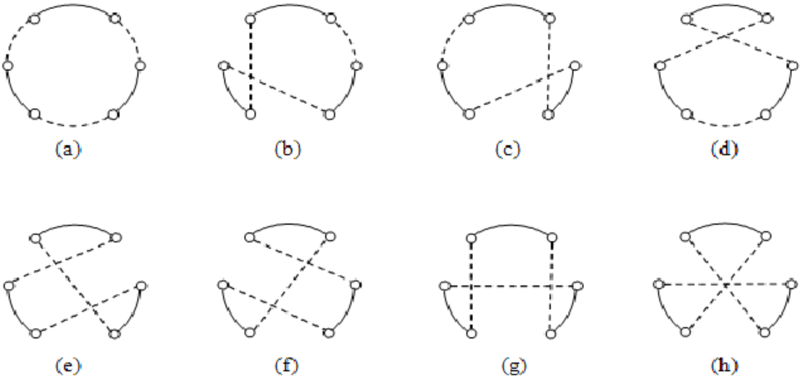
start with some small tour generated by a simple heuristic, then:

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

2-opt move



3-opt move



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
- there are other structures of graphs (point sets) that allow for finding easily optimal tours
  (so it is worthwhile to analyse input data beforehand...)

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.

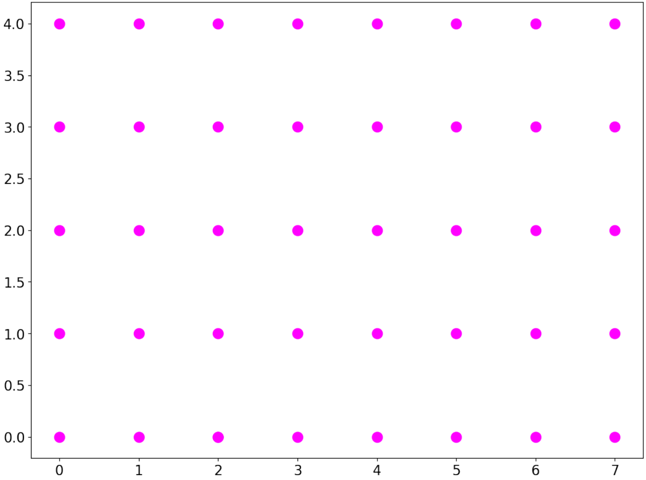
More on traveling salesperson problem

- 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

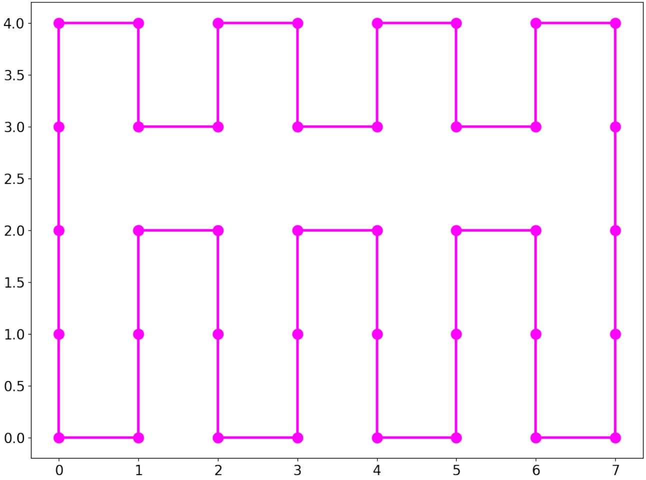
- In the 90's it was shown that eTSP can be solved in $\mathcal{O}(2^{O(\sqrt{n}\log n)})$.
- In the 10's this was improved to $\mathcal{O}(2^{\sqrt{n}})$, and with certain arguments that further improvement may be very unlikely.
- One recent result of complexity theory is that eTSP has a polynomial 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



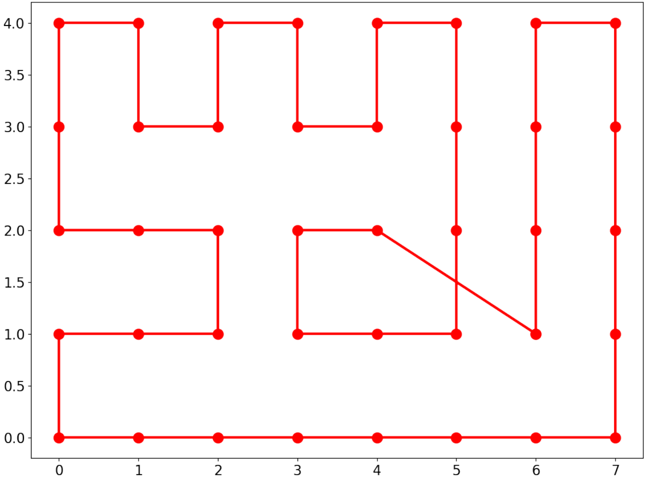
the cities distributed geographically

traveling salesperson problem: first impressions



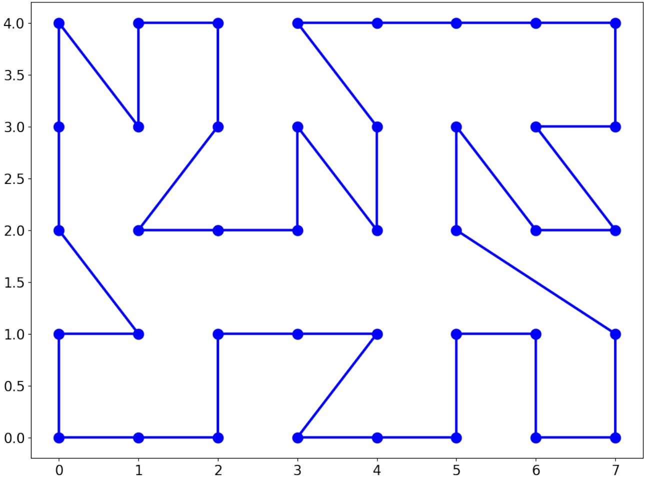
a best tour (trivial for this example): 0% gap

traveling salesperson problem: first impressions



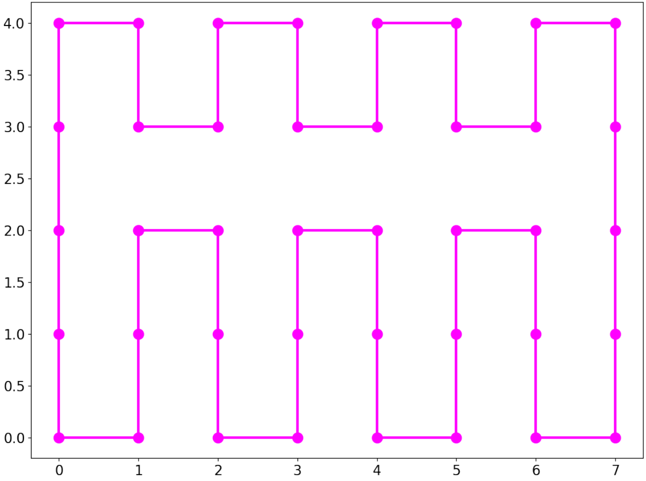
a closest-neighbor tour (with Monte Carlo): 3.09% gap

traveling salesperson problem: first impressions



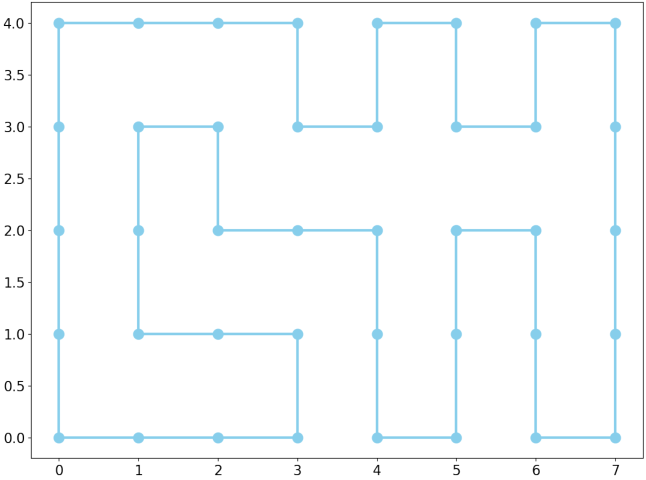
a pair-center tour: 14.46% gap

traveling salesperson problem: first impressions



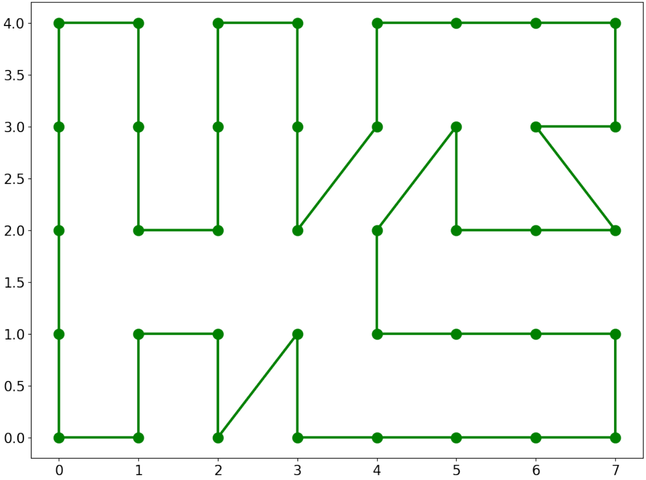
a best tour (trivial for this example): 0% gap

traveling salesperson problem: first impressions



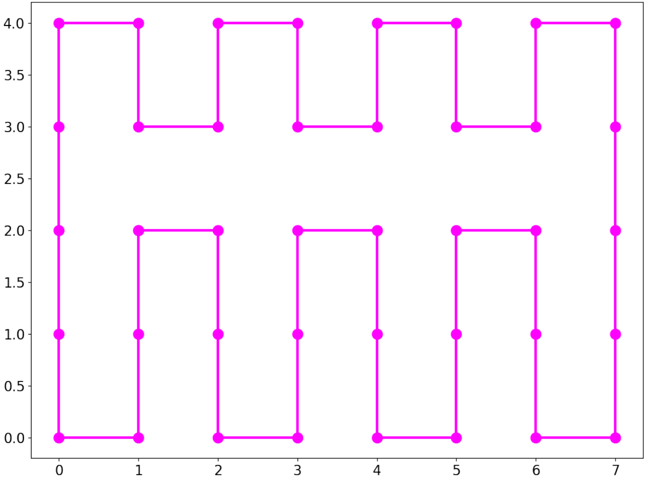
a quick tour (with Monte Carlo): 0.00% gap

traveling salesperson problem: first impressions



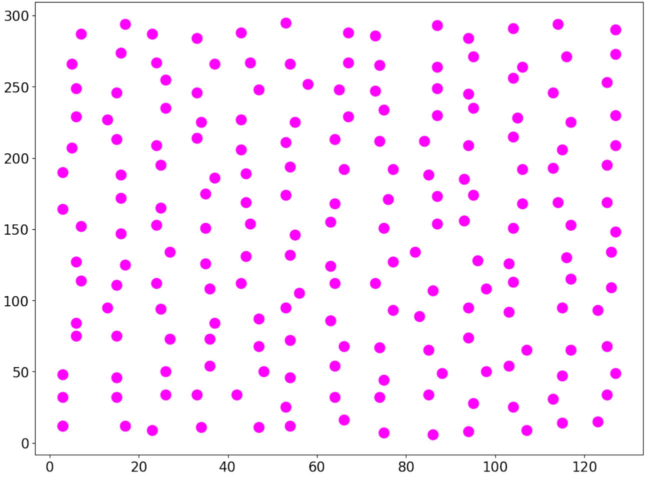
a genetic algorithm tour: 4.14% gap

traveling salesperson problem: first impressions



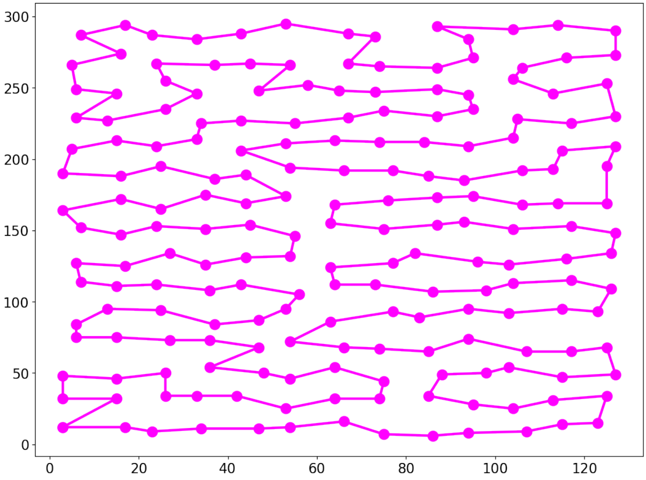
a best tour (trivial for this example): 0% gap

traveling salesperson problem: first impressions



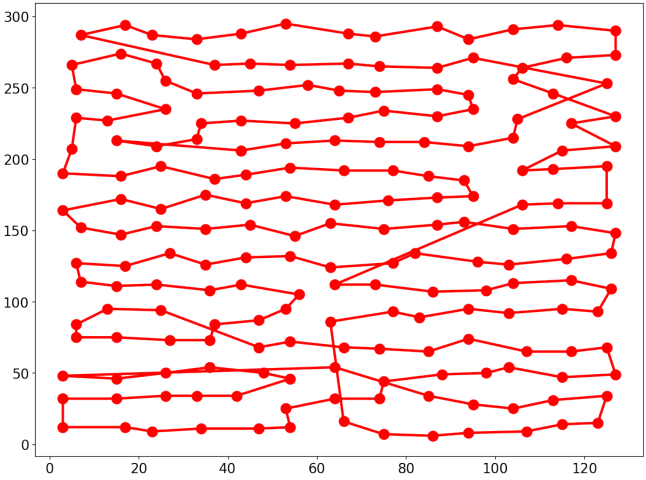
the locations distributed in the plane

traveling salesperson problem: first impressions



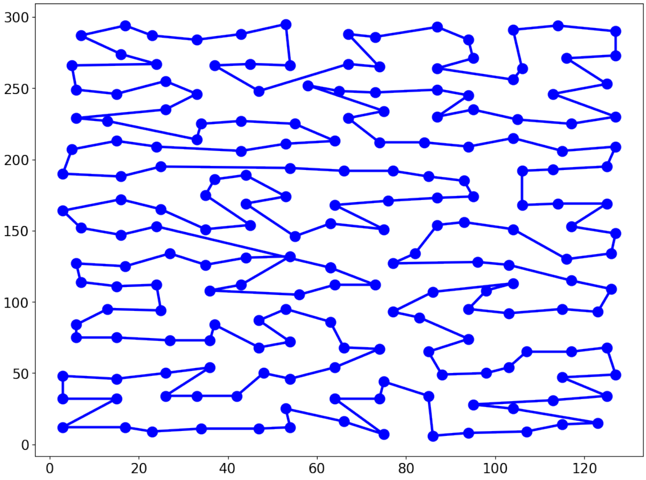
the best tour (known for this example): 0% gap

traveling salesperson problem: first impressions



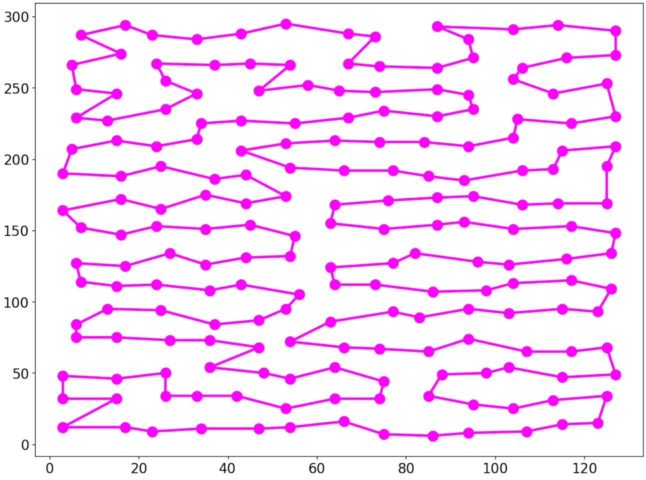
a closest-neighbor tour (with Monte Carlo): 10.23% gap

traveling salesperson problem: first impressions



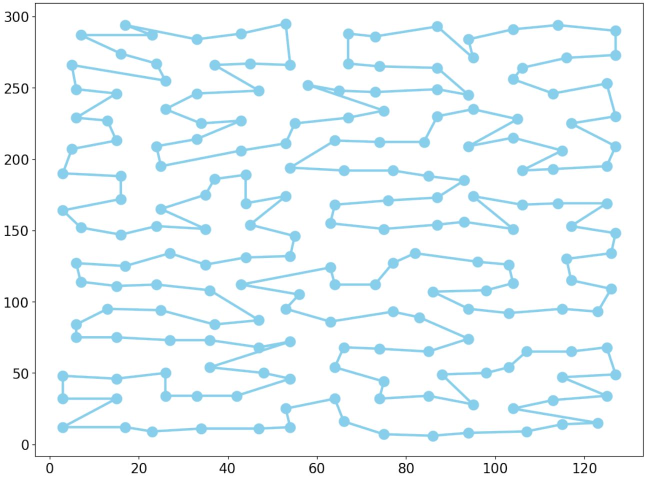
a pair-center tour: 13.93% gap

traveling salesperson problem: first impressions



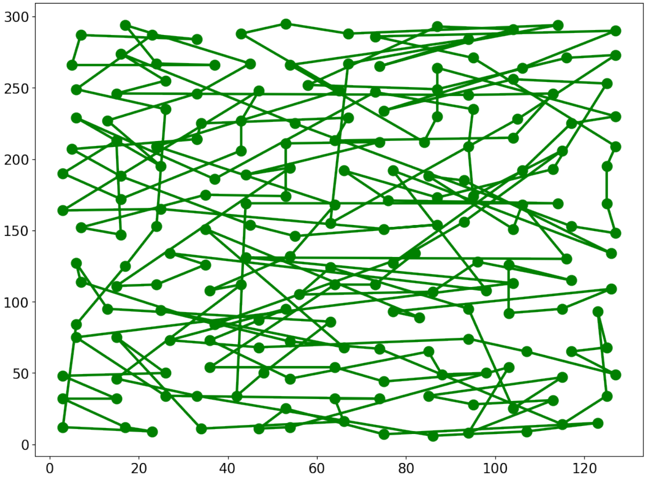
the best tour (known for this example): 0% gap

traveling salesperson problem: first impressions



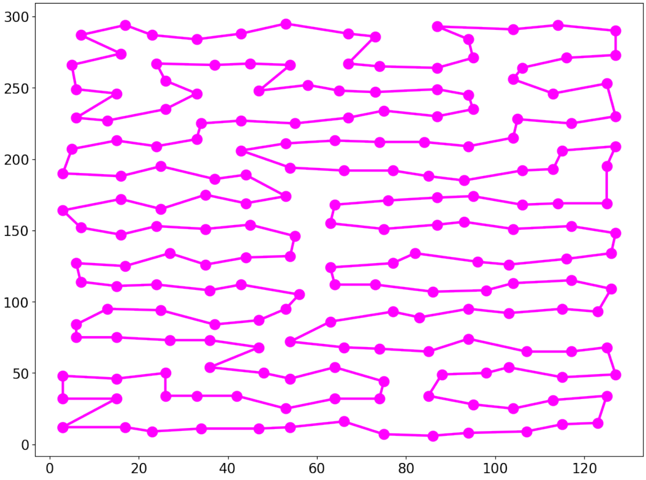
a quick tour (with Monte Carlo): 9.66% gap

traveling salesperson problem: first impressions



a genetic algorithm (GA) tour: 205.65% gap

traveling salesperson problem: first impressions



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

gaps for the above heuristics

problem	heuristic	gap
berlin52	closest neighbor tour	8.49
	quick tour	2.32
	pair-center tour	7.28
	genetic algorithm tour	8.37
	improved pair-center tour	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
	improved pair-center tour	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

Your goal: make genetic algorithm tour consistently better than pair-center tour or quick tour

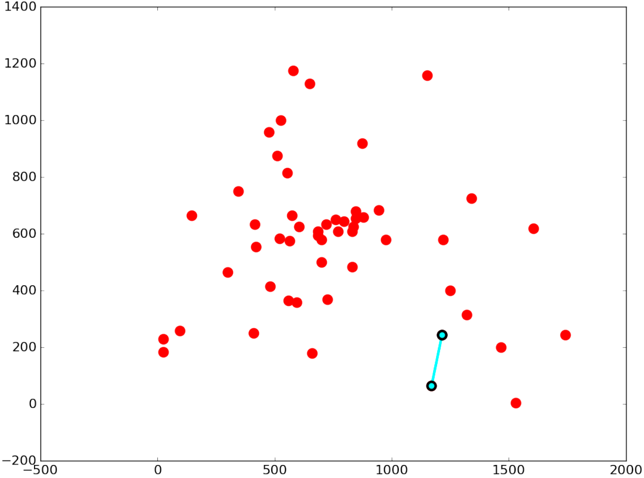
How does the closest-neighbor algorithms work?

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

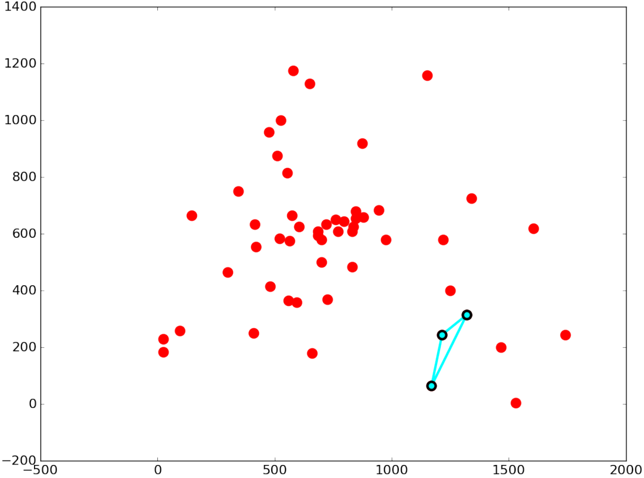
- select a random city
- while there are still unconnected cities
  - connect to the closest unconnected neighbor
  - use a random tie break
- connect the first with the last city

- 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 $0.5 \cdot \log n \cdot L_{opt}$

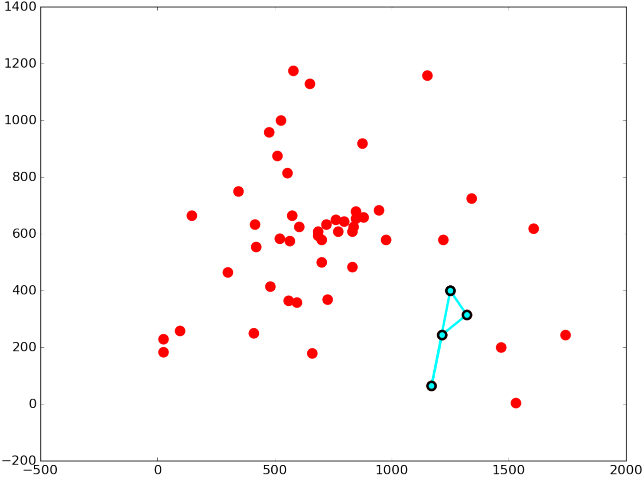
animation of the closest-neighbor algorithm



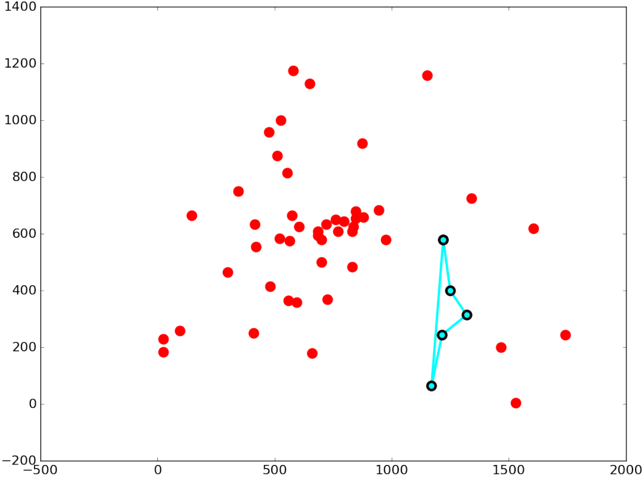
animation of the closest-neighbor algorithm



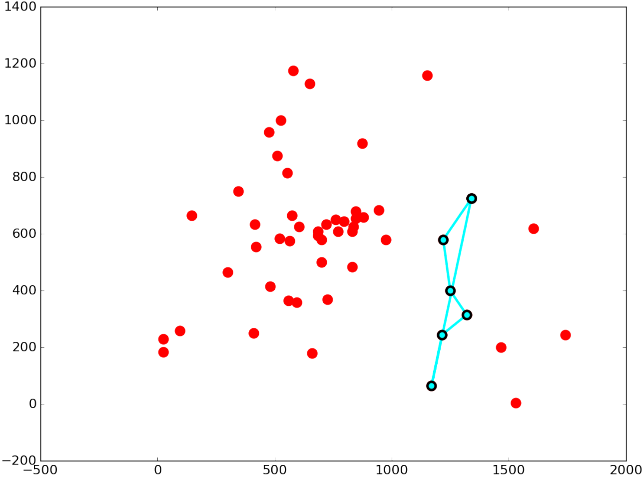
animation of the closest-neighbor algorithm



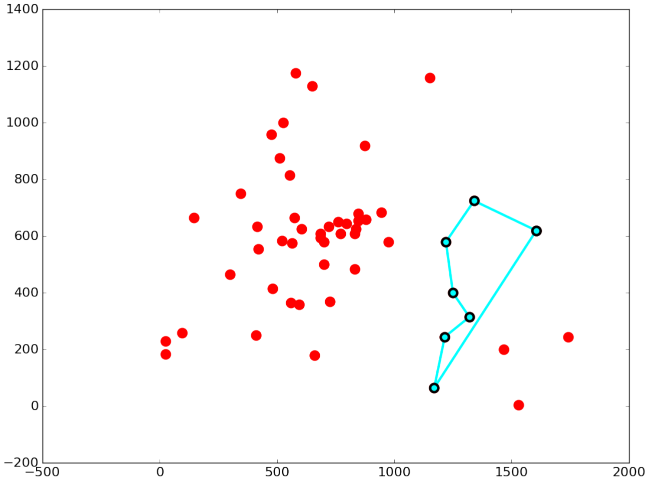
animation of the closest-neighbor algorithm



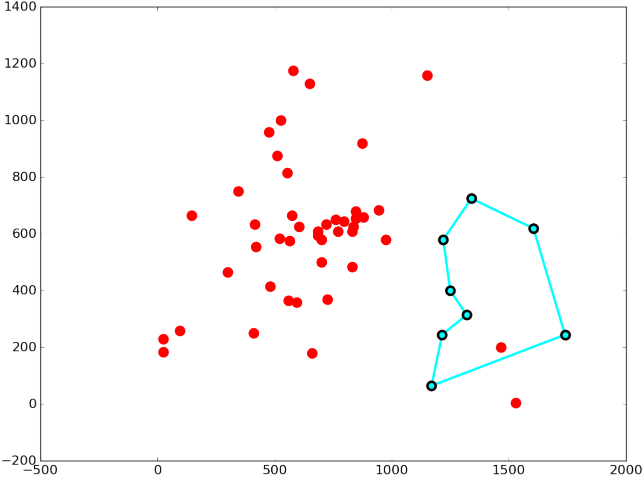
animation of the closest-neighbor algorithm



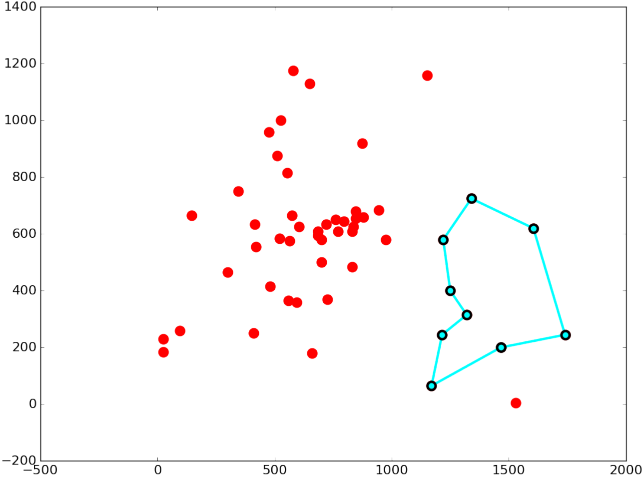
animation of the closest-neighbor algorithm



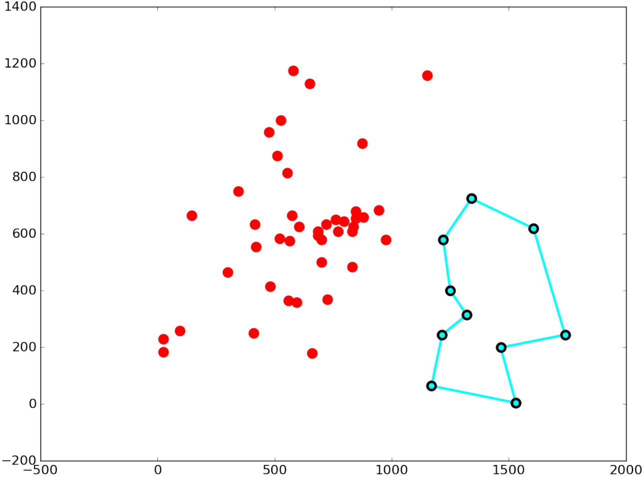
animation of the closest-neighbor algorithm



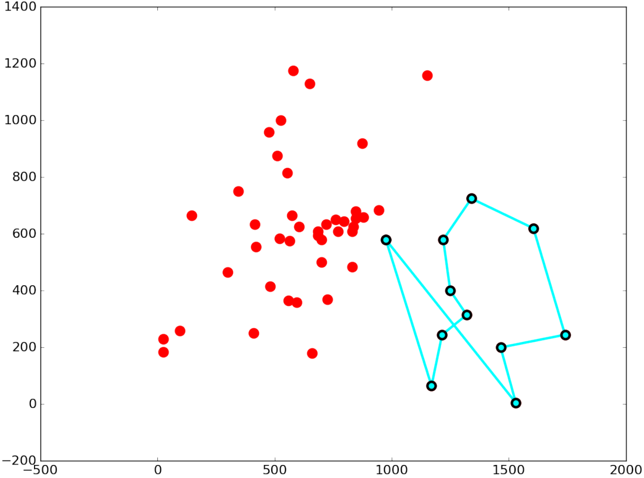
animation of the closest-neighbor algorithm



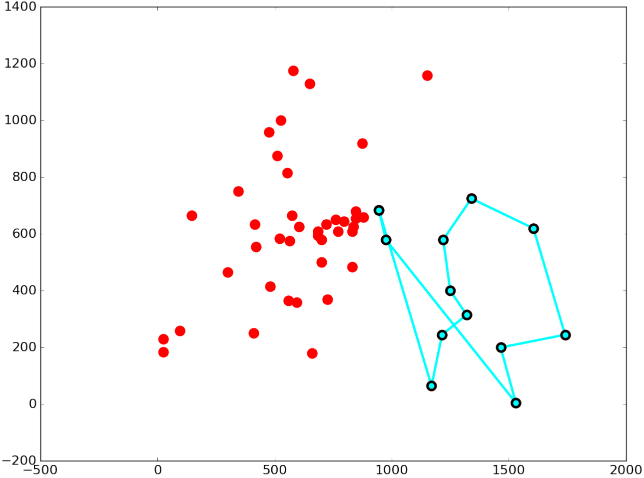
animation of the closest-neighbor algorithm



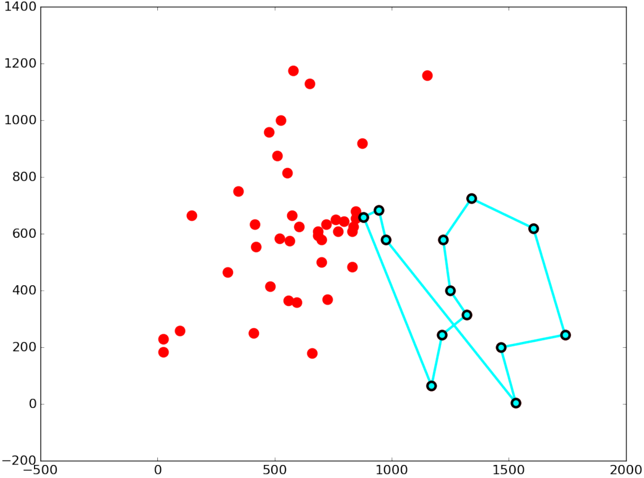
animation of the closest-neighbor algorithm



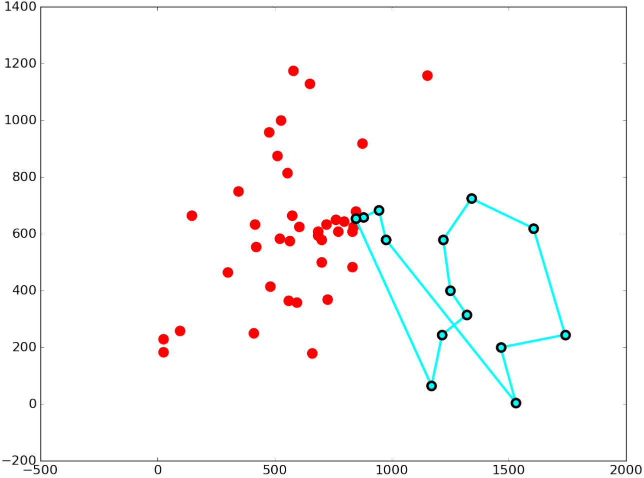
animation of the closest-neighbor algorithm



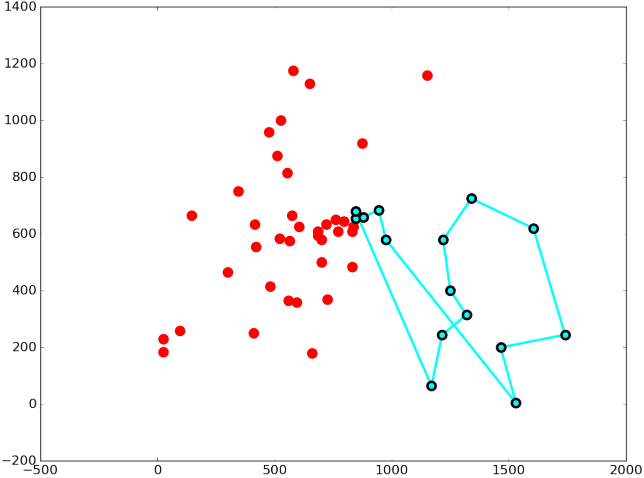
animation of the closest-neighbor algorithm



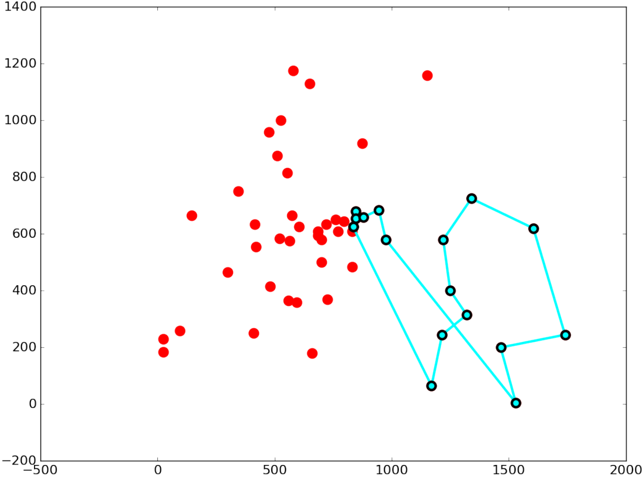
animation of the closest-neighbor algorithm



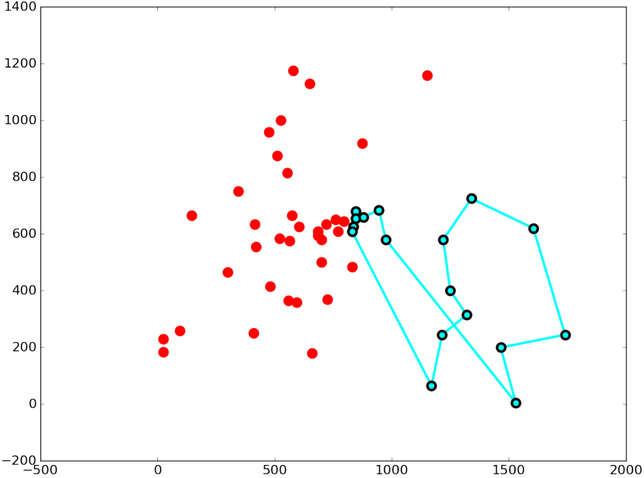
animation of the closest-neighbor algorithm



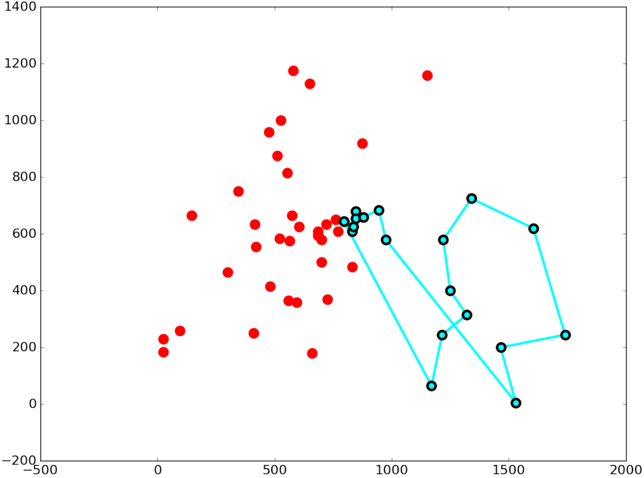
animation of the closest-neighbor algorithm



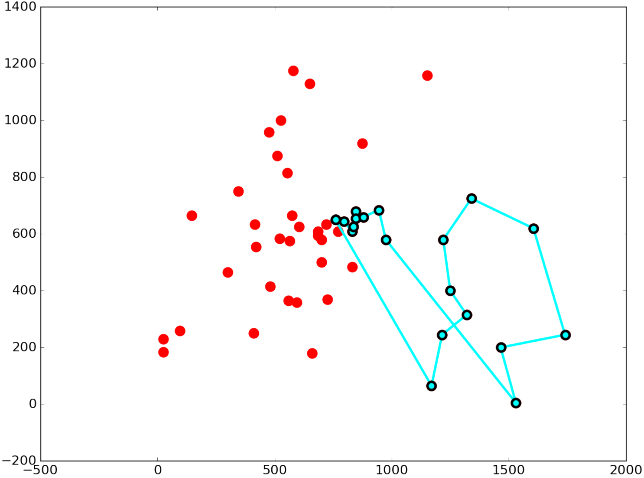
animation of the closest-neighbor algorithm



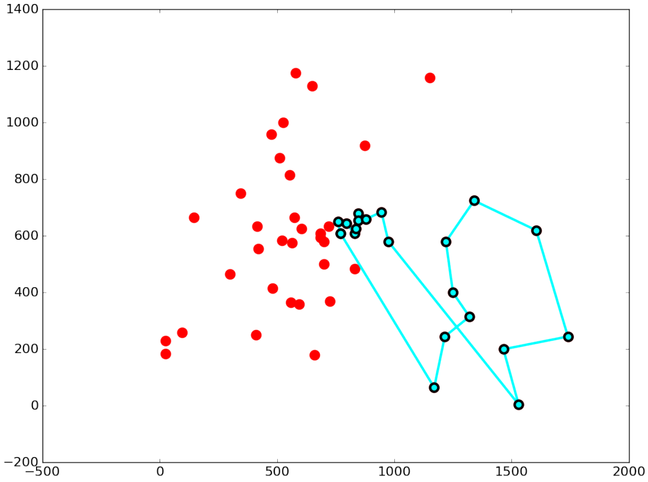
animation of the closest-neighbor algorithm



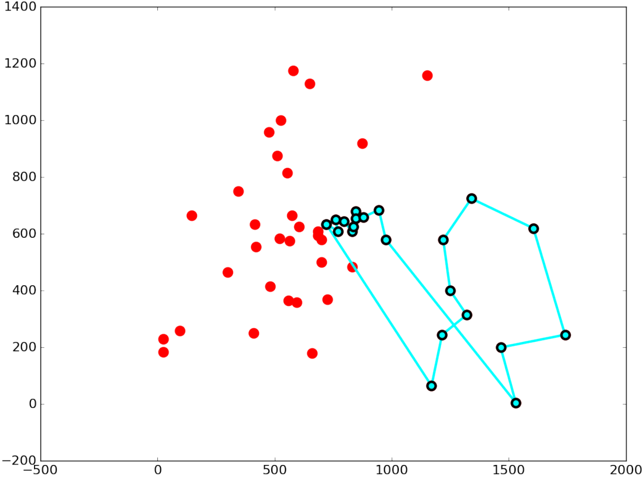
animation of the closest-neighbor algorithm



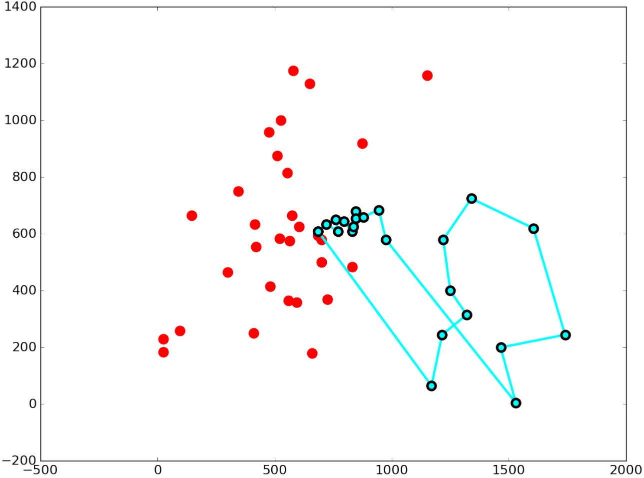
animation of the closest-neighbor algorithm



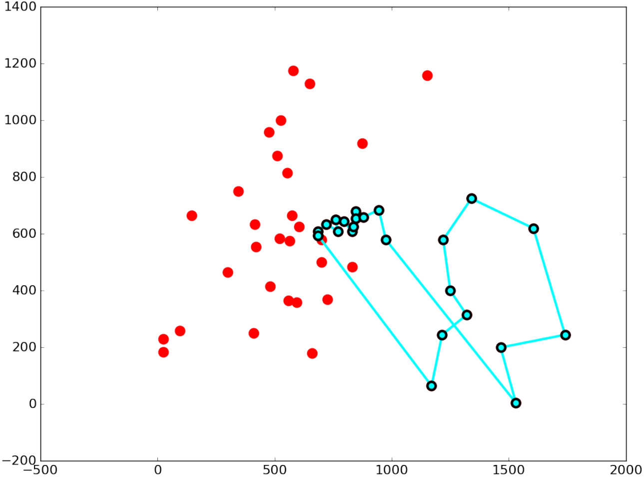
animation of the closest-neighbor algorithm



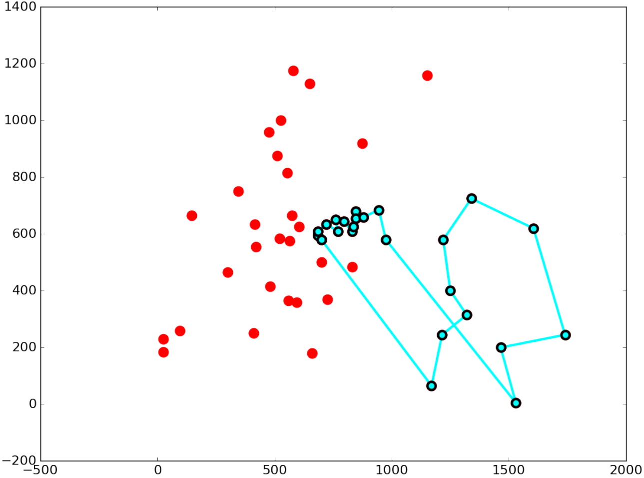
animation of the closest-neighbor algorithm



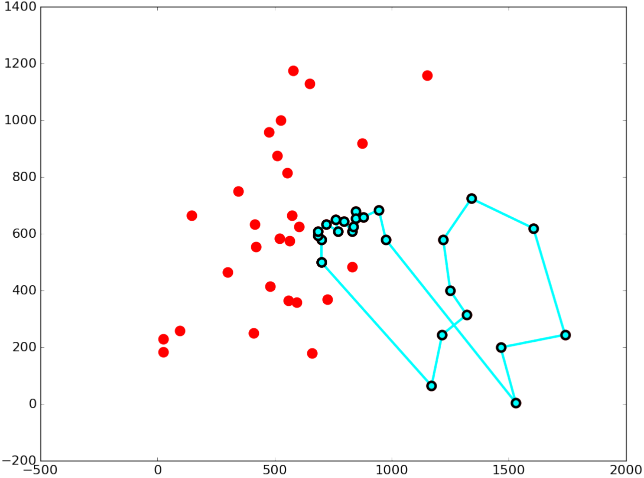
animation of the closest-neighbor algorithm



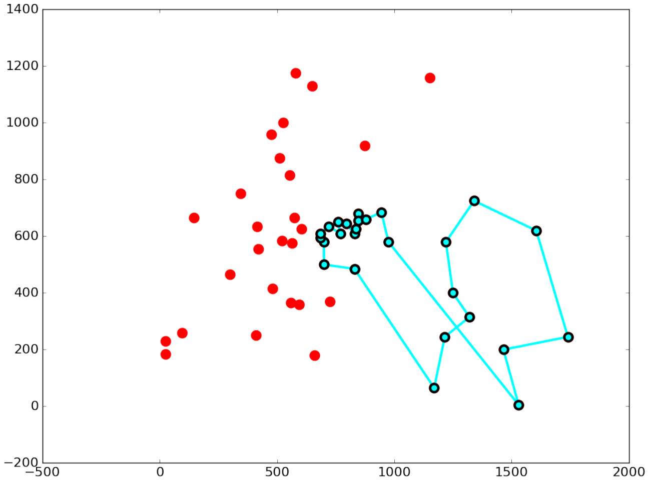
animation of the closest-neighbor algorithm



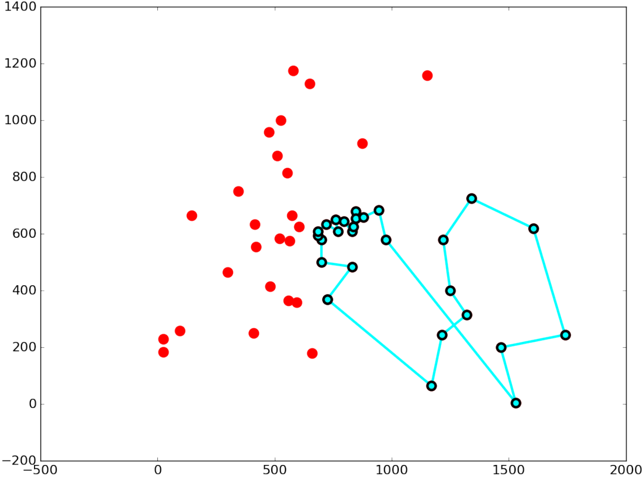
animation of the closest-neighbor algorithm



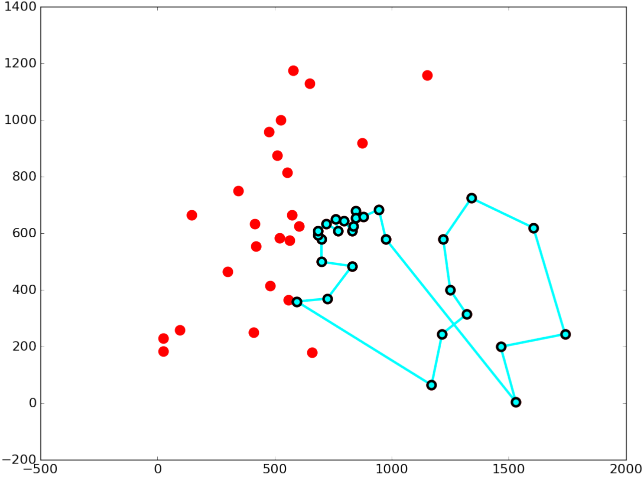
animation of the closest-neighbor algorithm



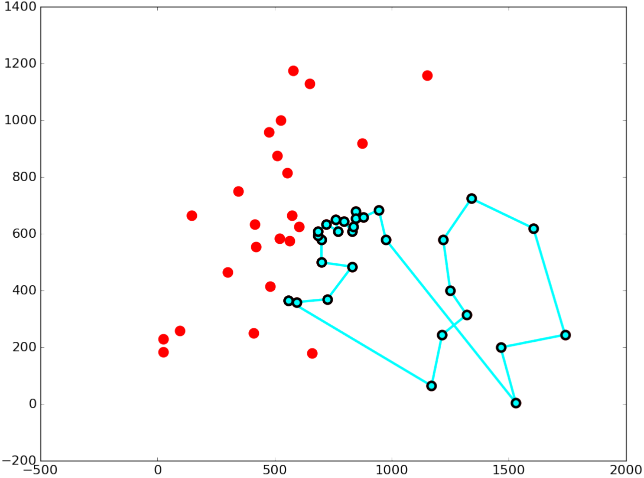
animation of the closest-neighbor algorithm



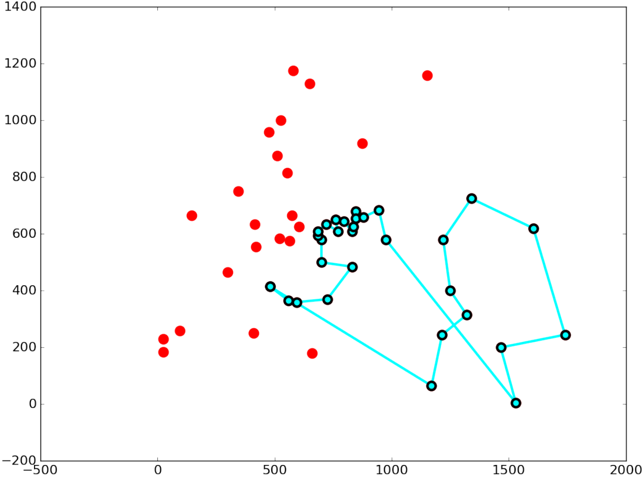
animation of the closest-neighbor algorithm



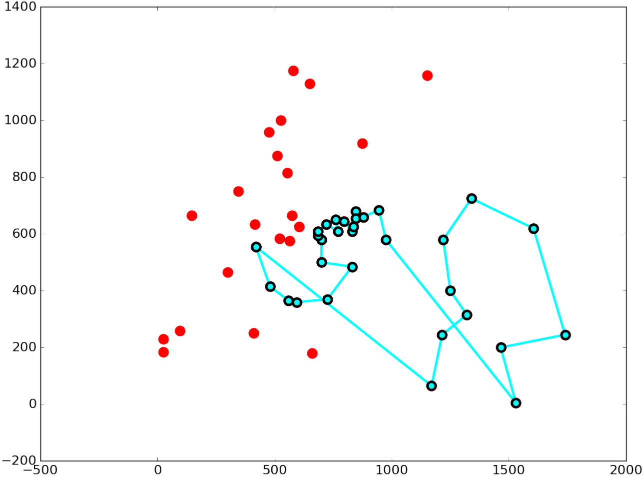
animation of the closest-neighbor algorithm



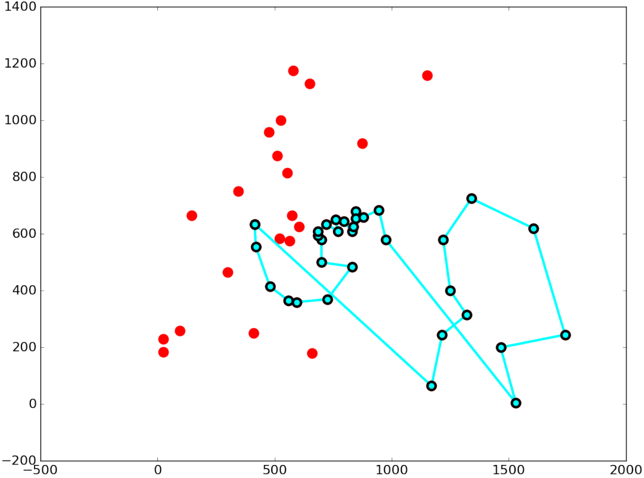
animation of the closest-neighbor algorithm



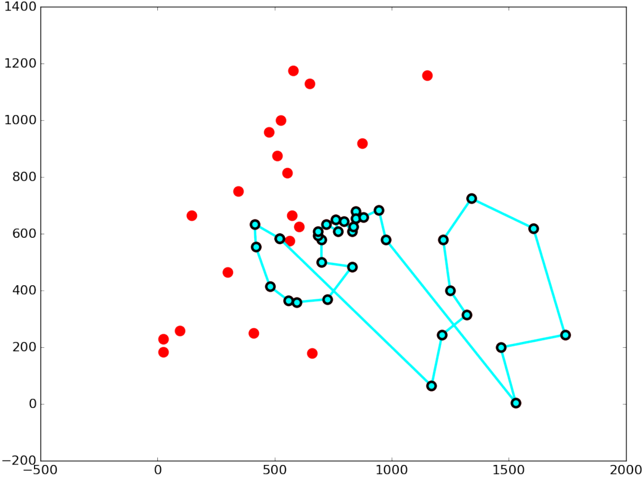
animation of the closest-neighbor algorithm



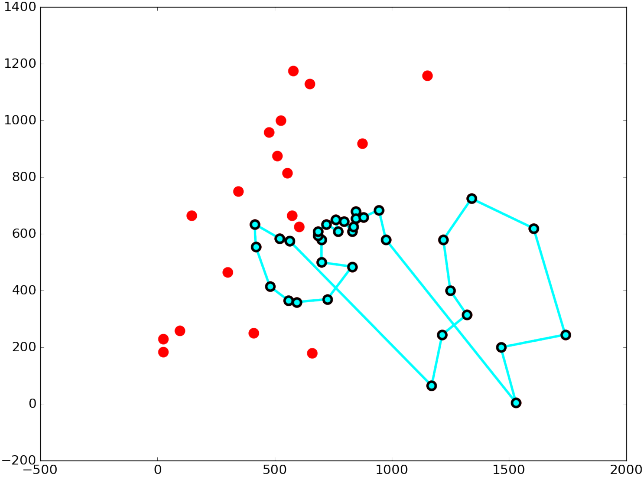
animation of the closest-neighbor algorithm



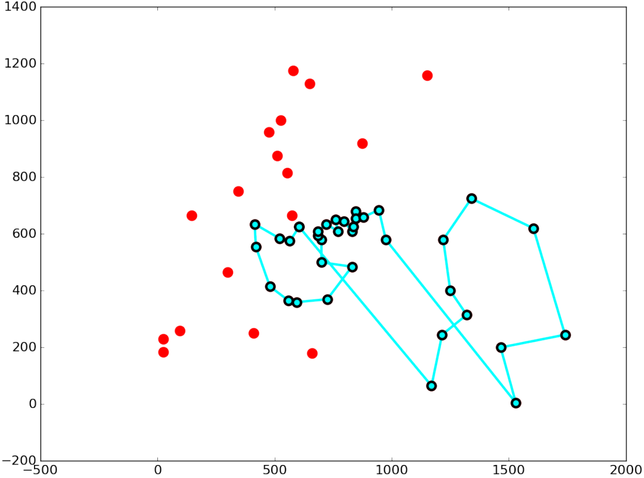
animation of the closest-neighbor algorithm



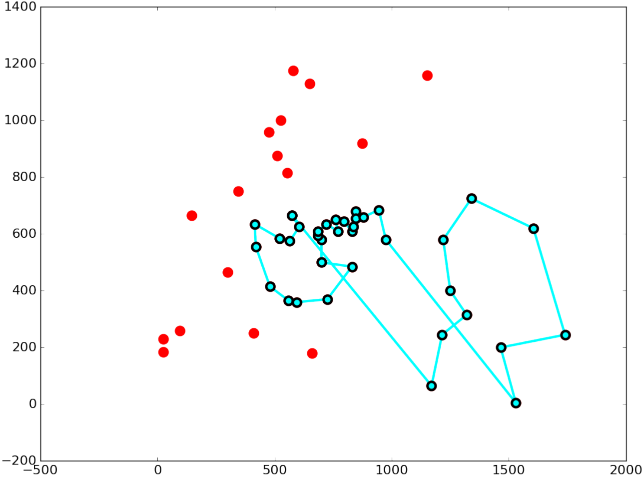
animation of the closest-neighbor algorithm



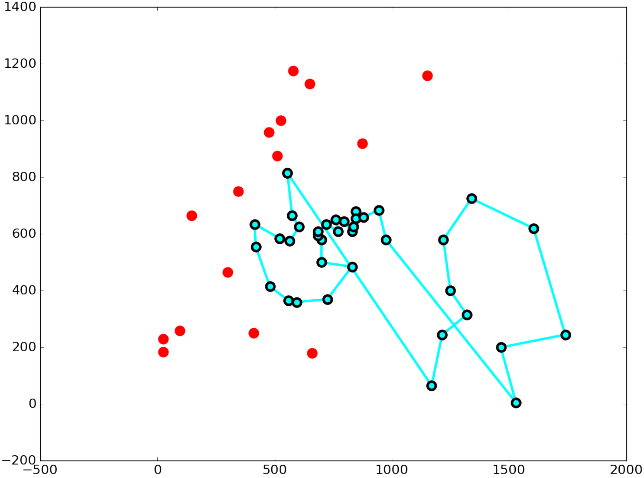
animation of the closest-neighbor algorithm



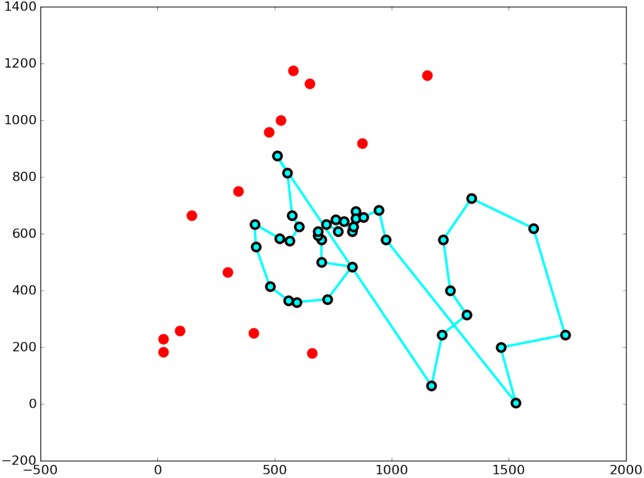
animation of the closest-neighbor algorithm



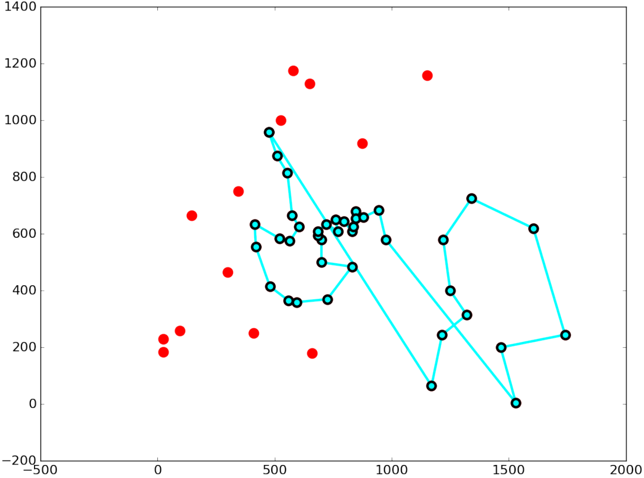
animation of the closest-neighbor algorithm



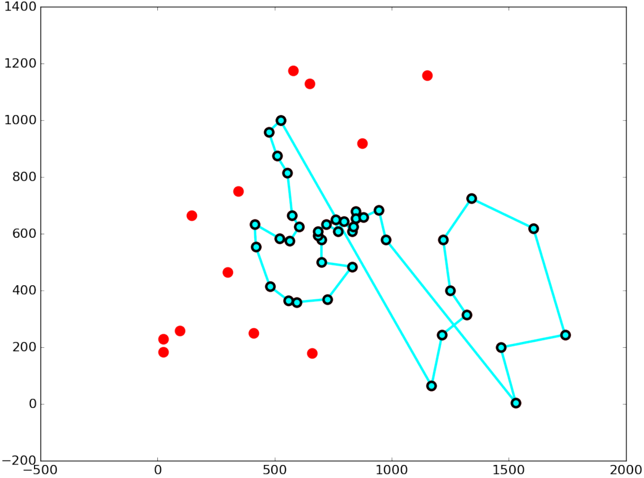
animation of the closest-neighbor algorithm



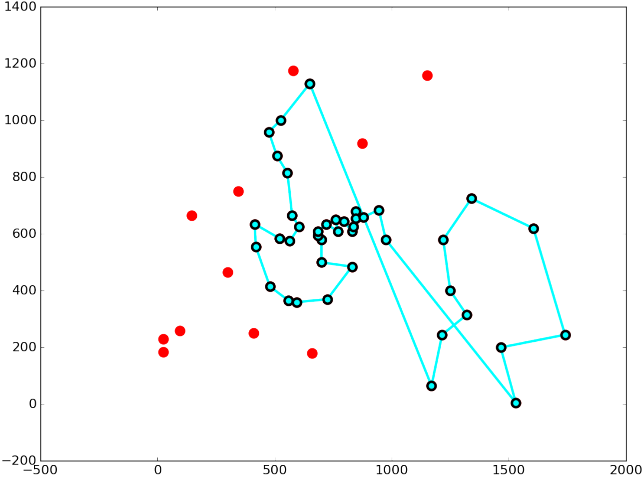
animation of the closest-neighbor algorithm



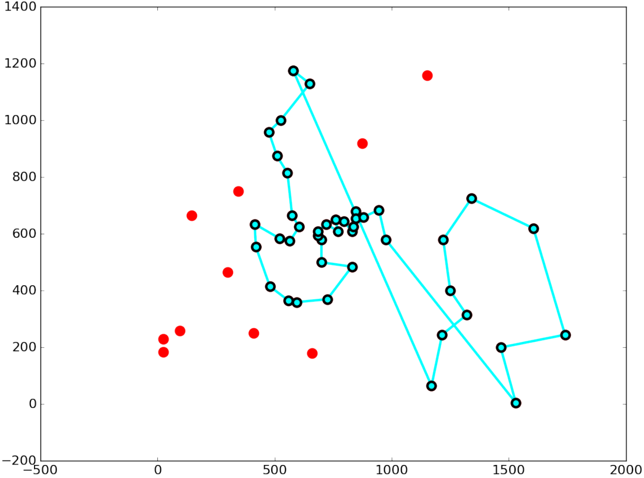
animation of the closest-neighbor algorithm



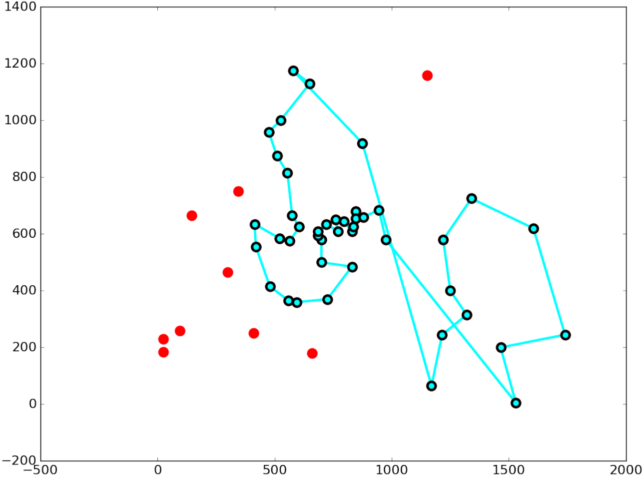
animation of the closest-neighbor algorithm



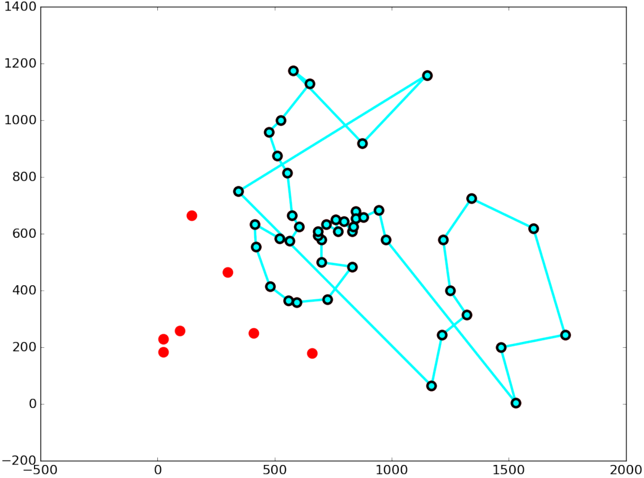
animation of the closest-neighbor algorithm



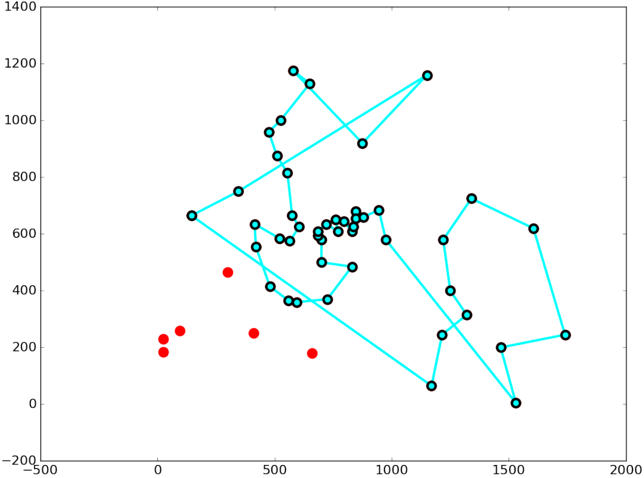
animation of the closest-neighbor algorithm



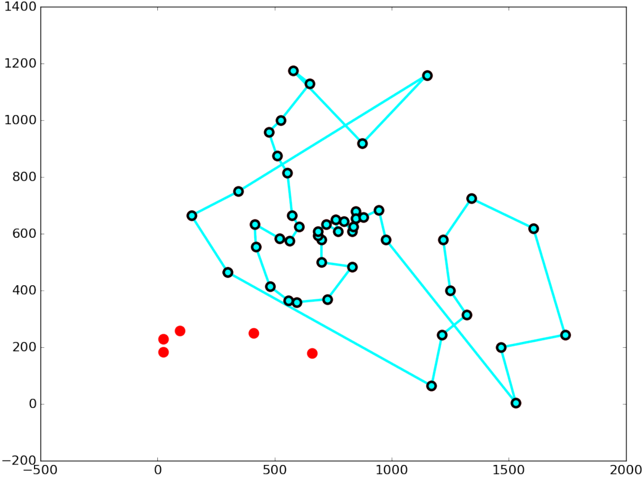
animation of the closest-neighbor algorithm



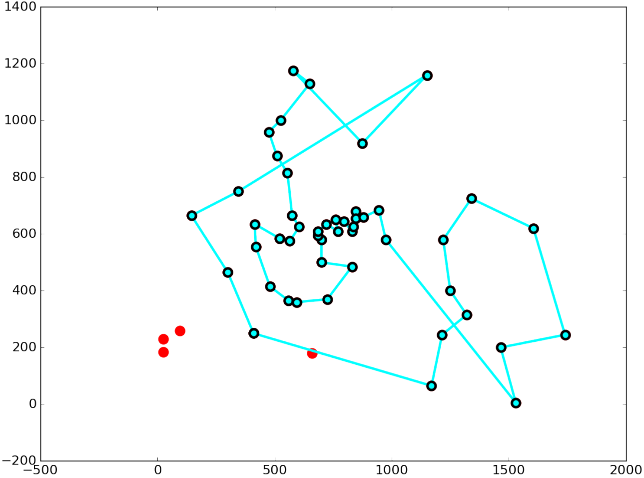
animation of the closest-neighbor algorithm



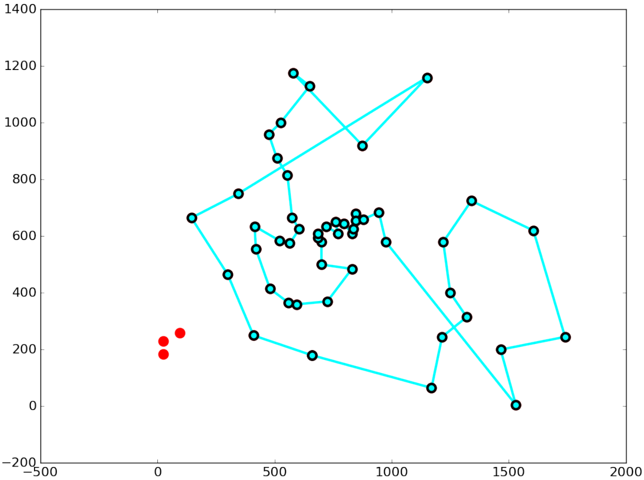
animation of the closest-neighbor algorithm



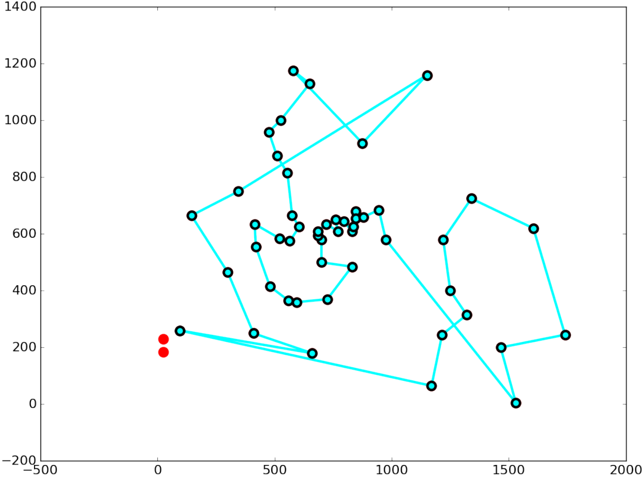
animation of the closest-neighbor algorithm



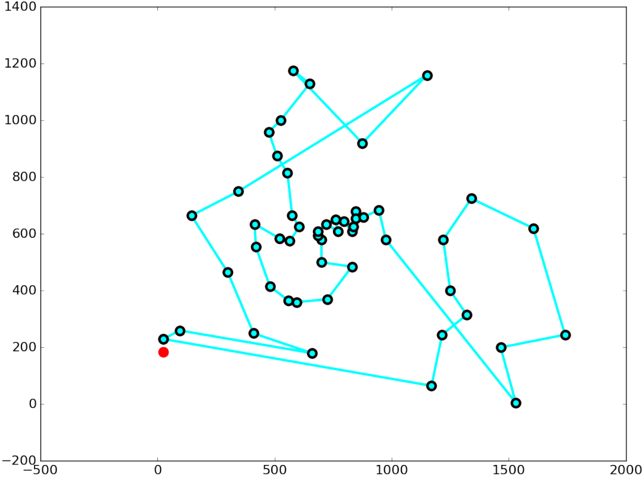
animation of the closest-neighbor algorithm



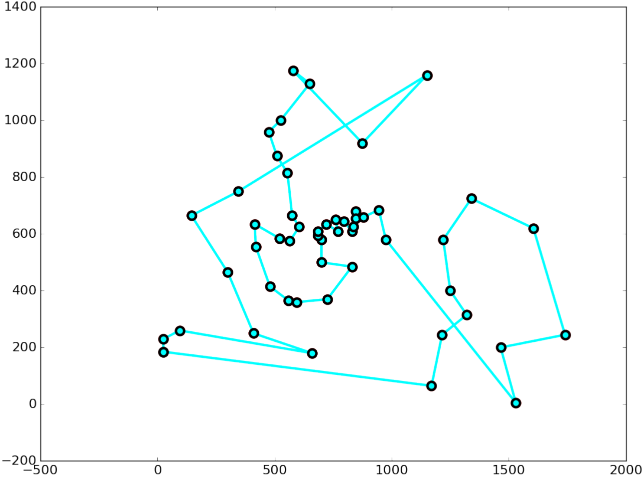
animation of the closest-neighbor algorithm



animation of the closest-neighbor algorithm



animation of the closest-neighbor algorithm



How does the quick tour algorithm work?

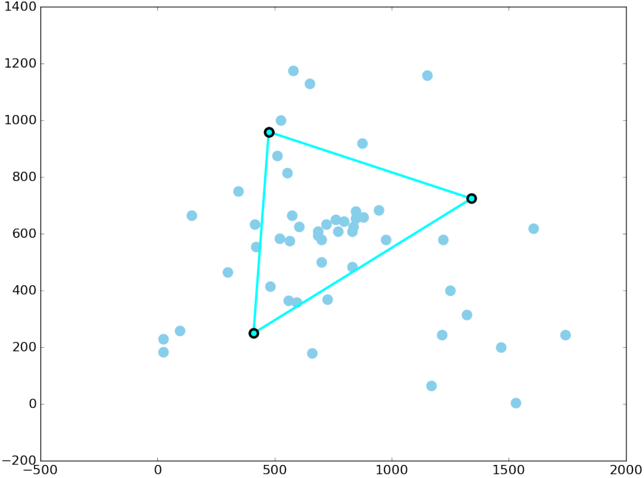
The quick tour algorithm (also known as *random insertion approach*) is a probabilistic greedy algorithm:

- 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

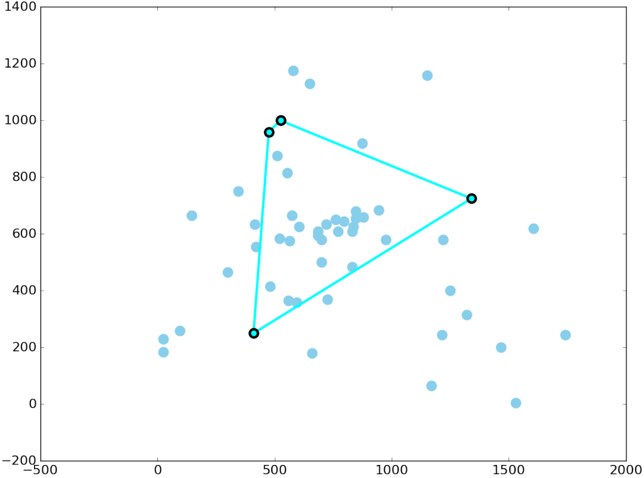
- runtime is in $\mathcal{O}(n^2)$,
- the random version can be run in Monte Carlo fashion keeping the shortest tour

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

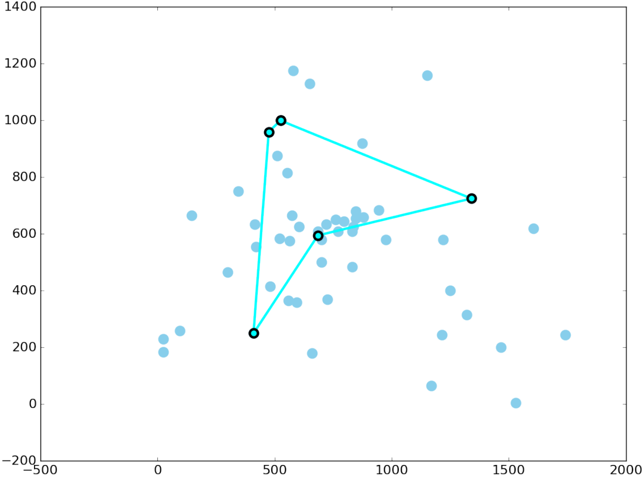
animation of the quick tour algorithm



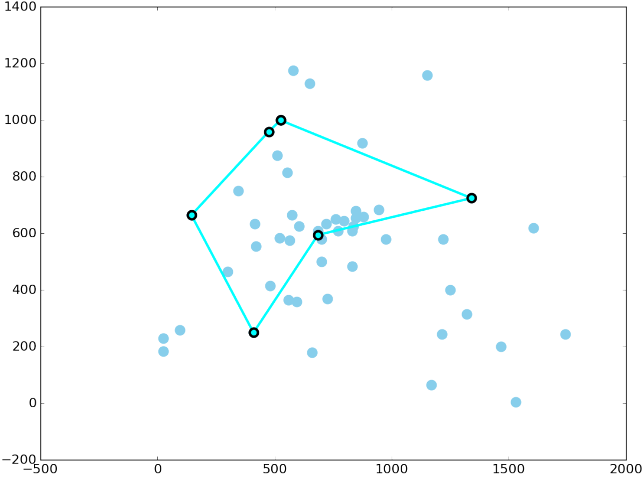
animation of the quick tour algorithm



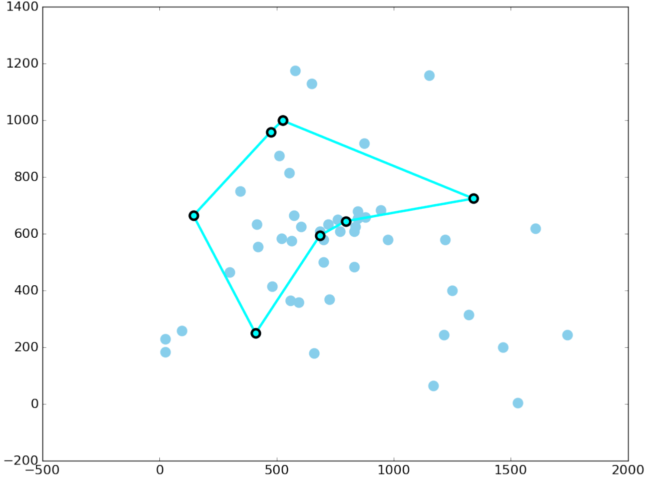
animation of the quick tour algorithm



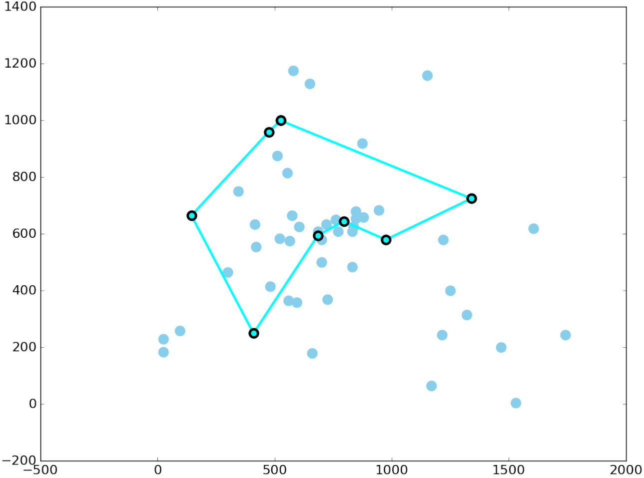
animation of the quick tour algorithm



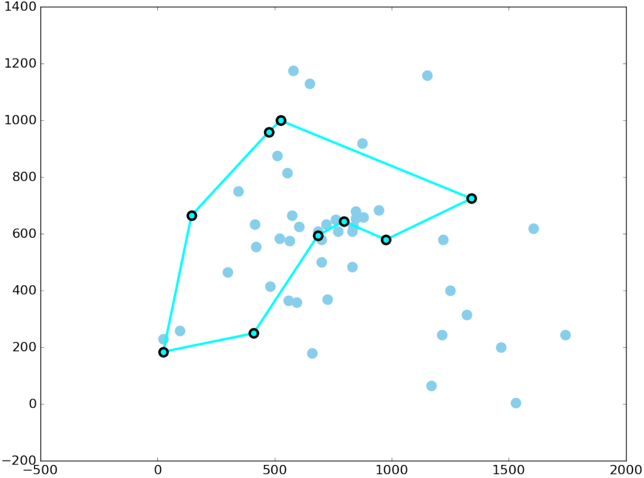
animation of the quick tour algorithm



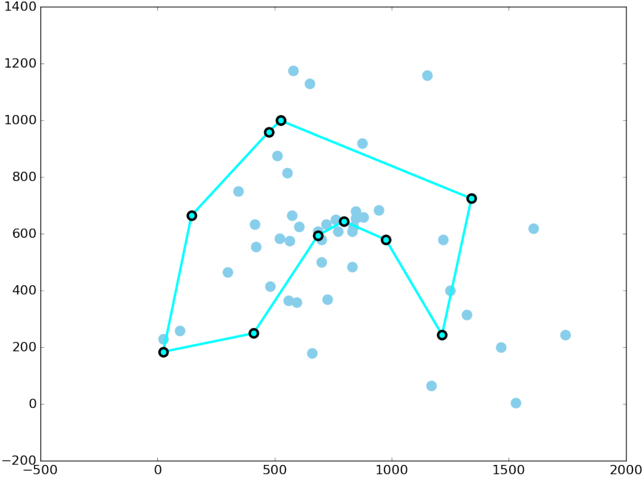
animation of the quick tour algorithm



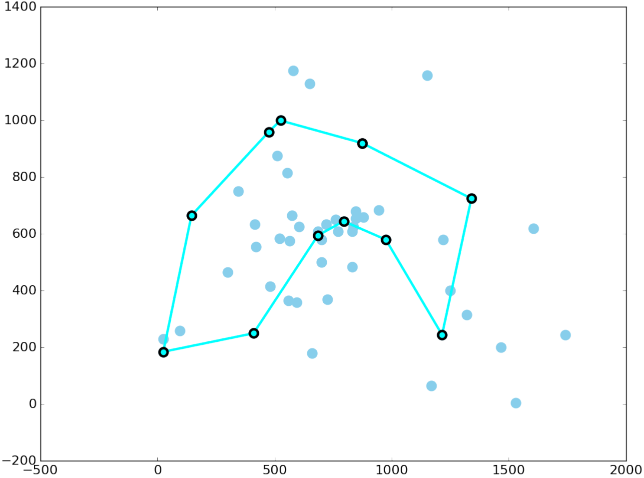
animation of the quick tour algorithm



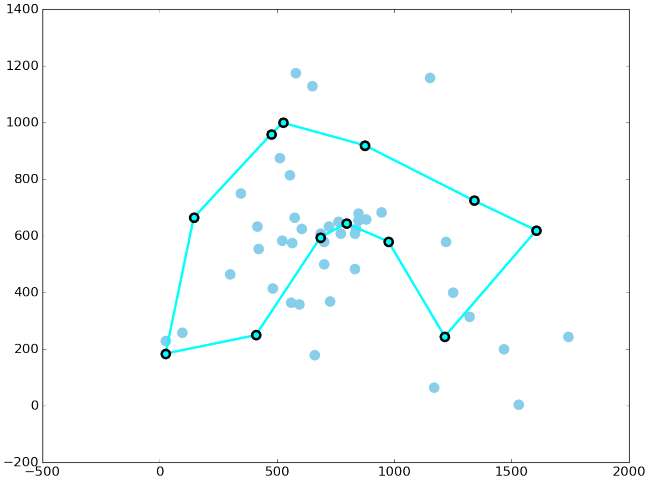
animation of the quick tour algorithm



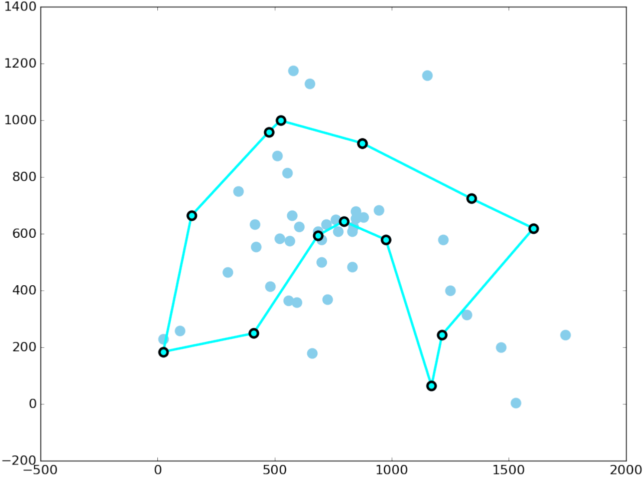
animation of the quick tour algorithm



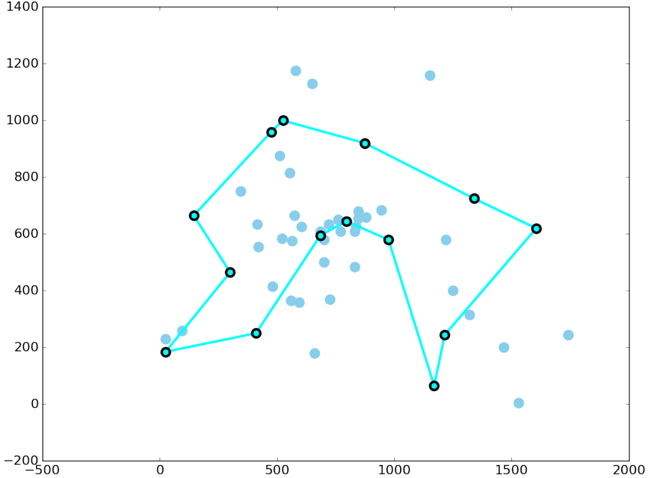
animation of the quick tour algorithm



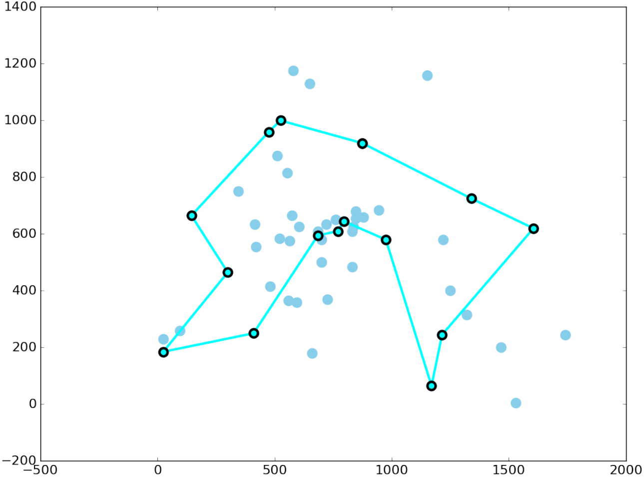
animation of the quick tour algorithm



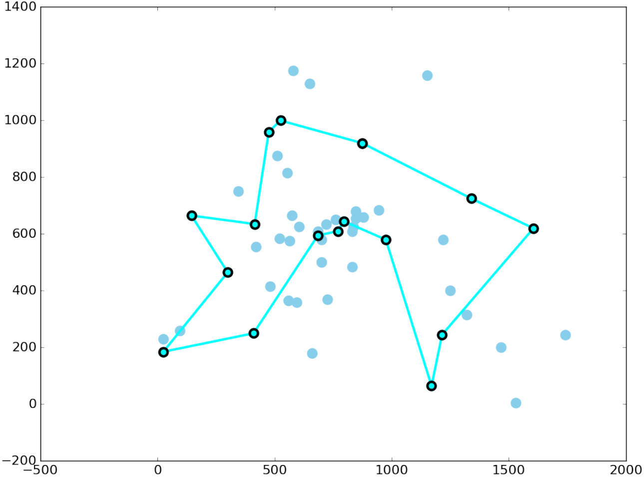
animation of the quick tour algorithm

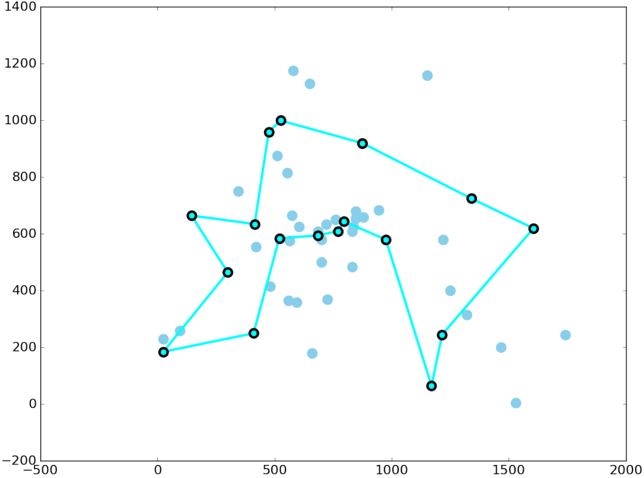


animation of the quick tour algorithm

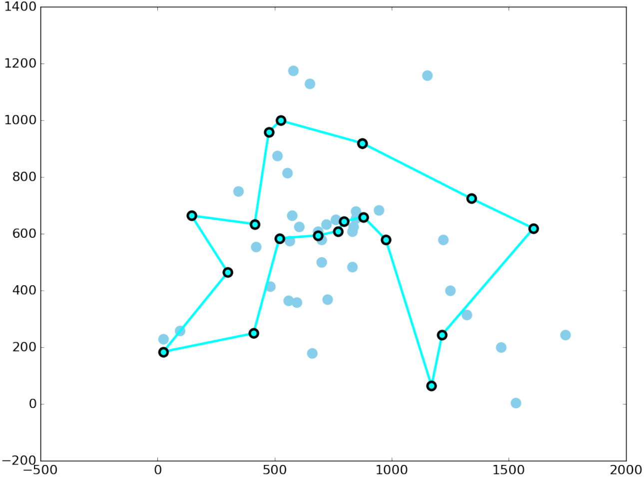


animation of the quick tour algorithm

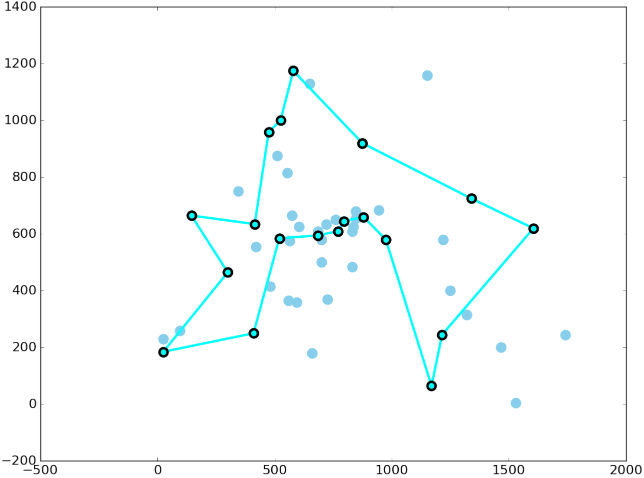



1400
1200
1000
800
600
400
200
0
−200
−500
0
500
1000
1500
2000

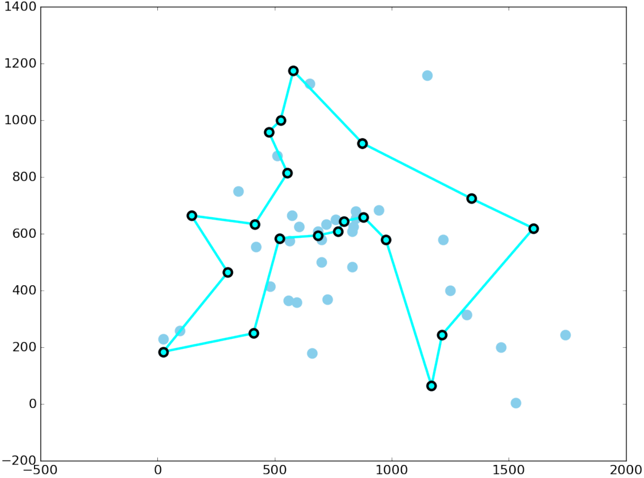
animation of the quick tour algorithm



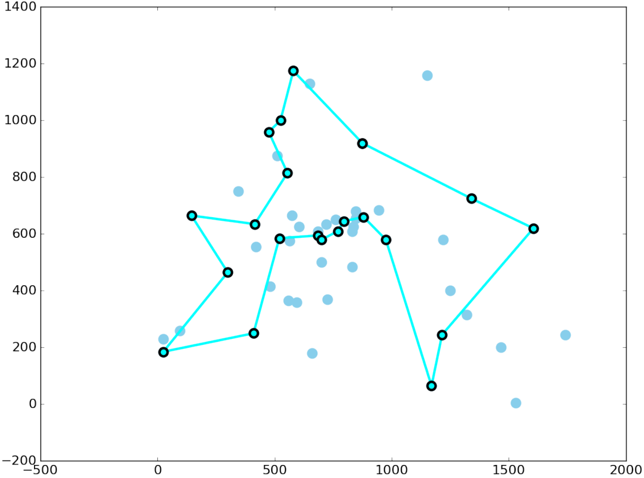
animation of the quick tour algorithm



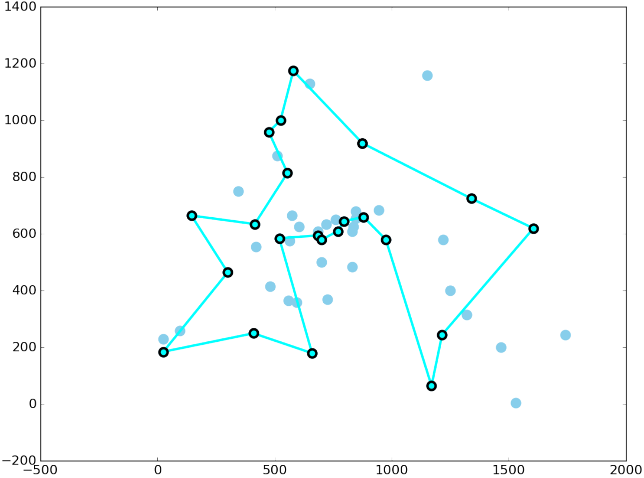
animation of the quick tour algorithm



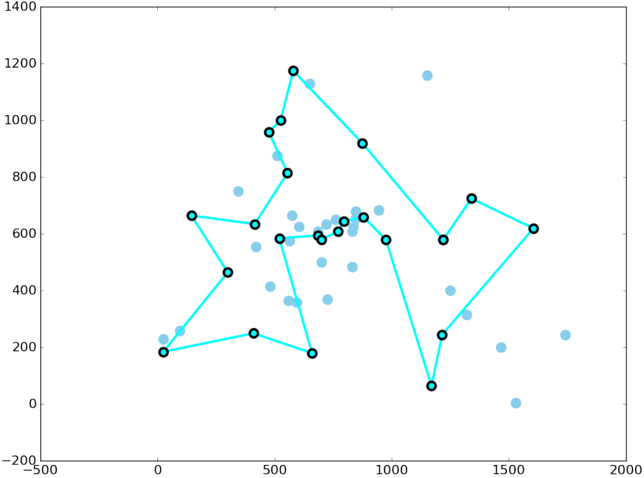
animation of the quick tour algorithm



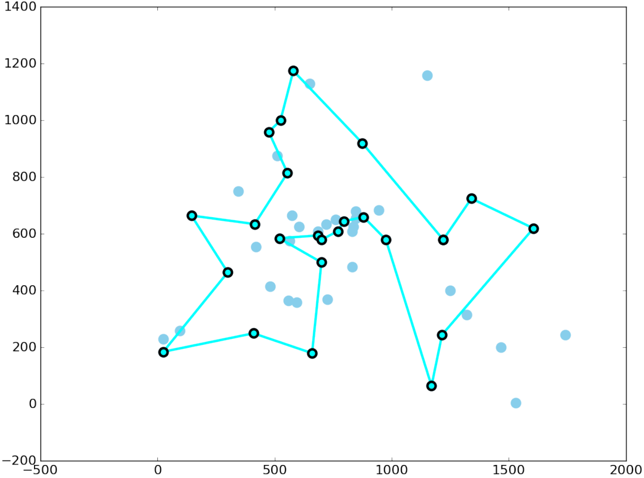
animation of the quick tour algorithm



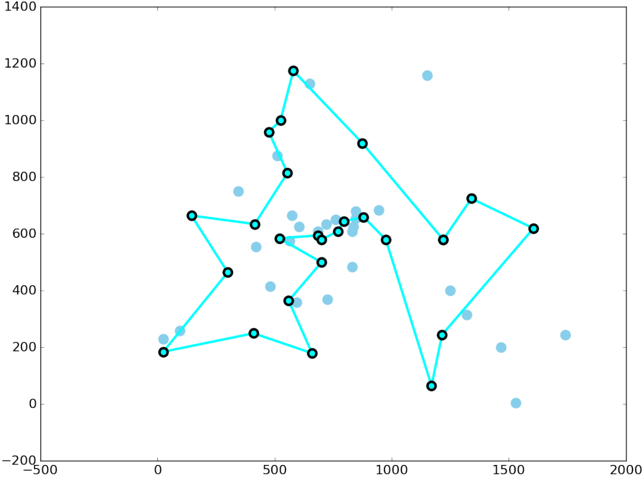
animation of the quick tour algorithm



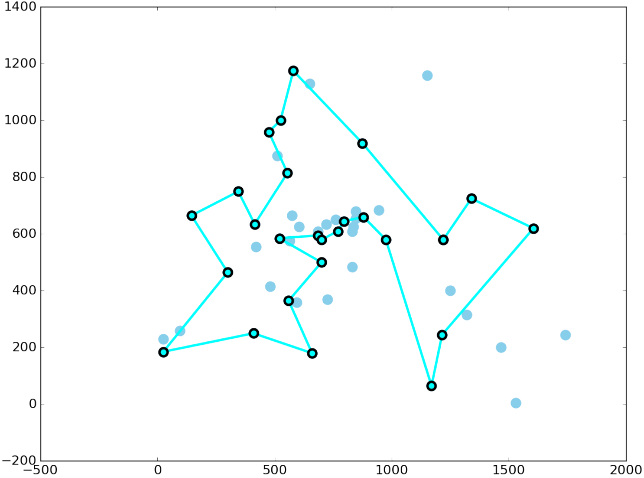
animation of the quick tour algorithm



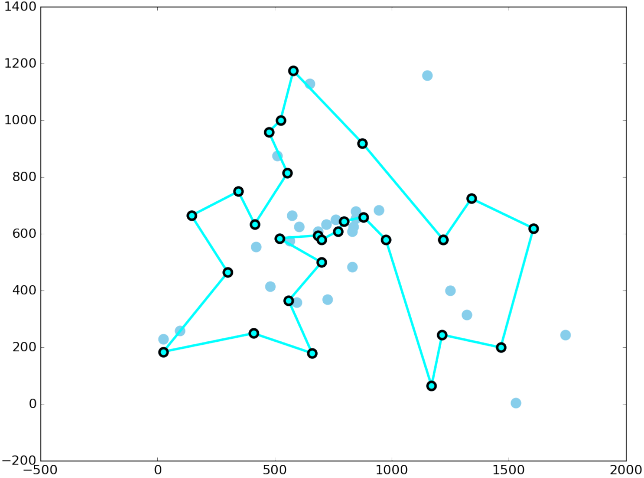
animation of the quick tour algorithm



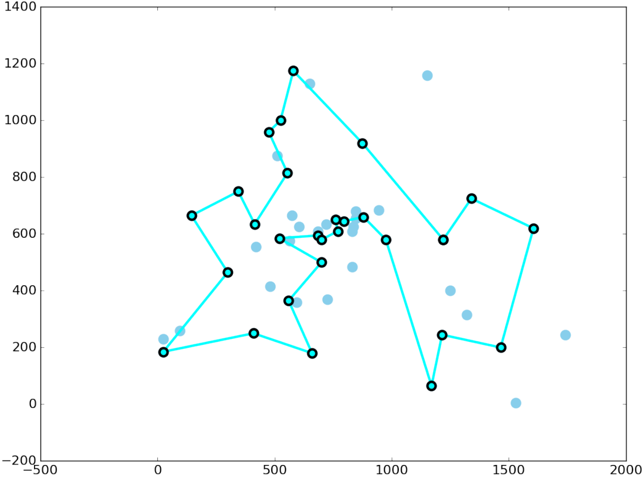
animation of the quick tour algorithm



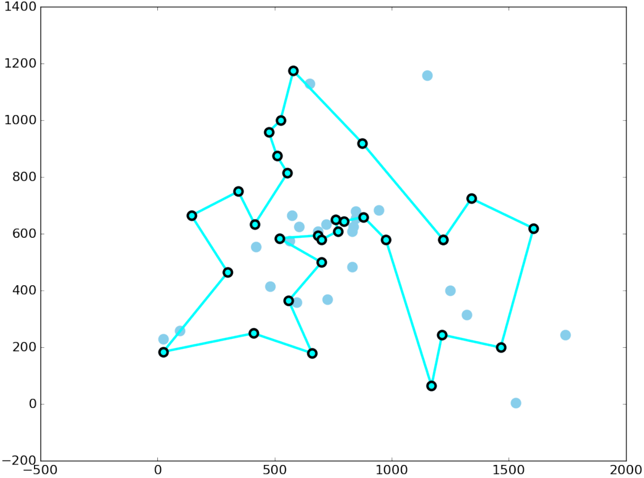
animation of the quick tour algorithm



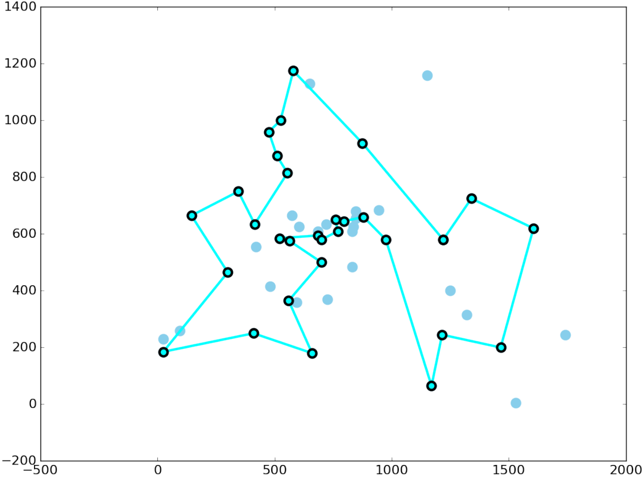
animation of the quick tour algorithm



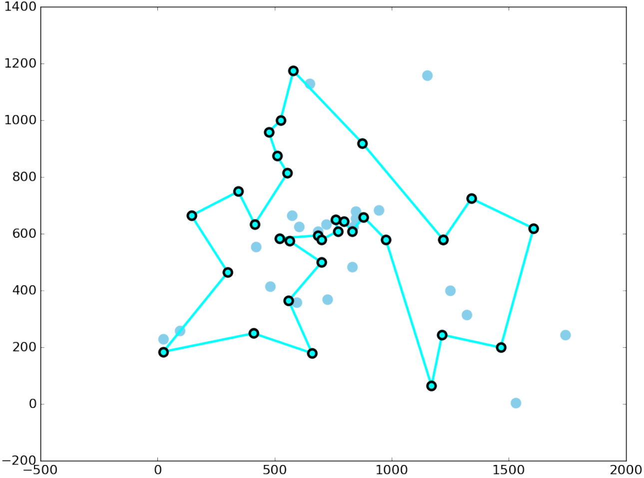
animation of the quick tour algorithm



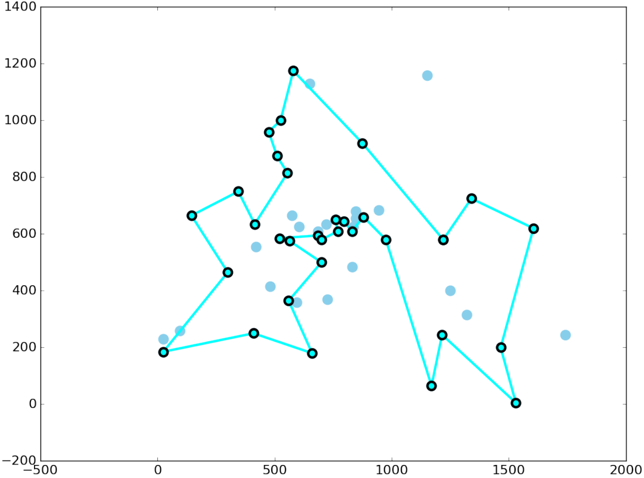
animation of the quick tour algorithm



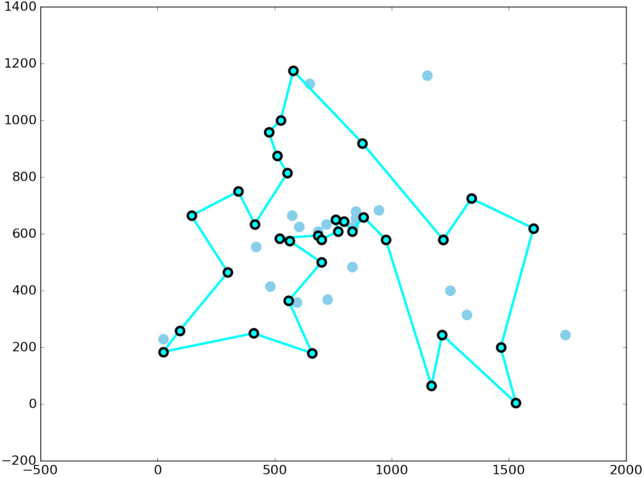
animation of the quick tour algorithm



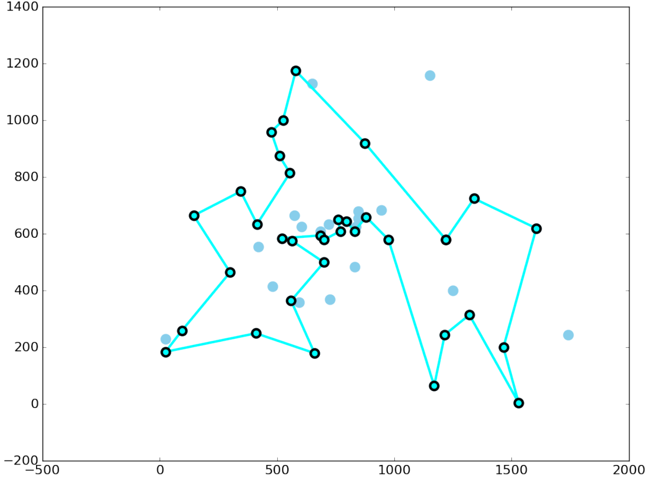
animation of the quick tour algorithm



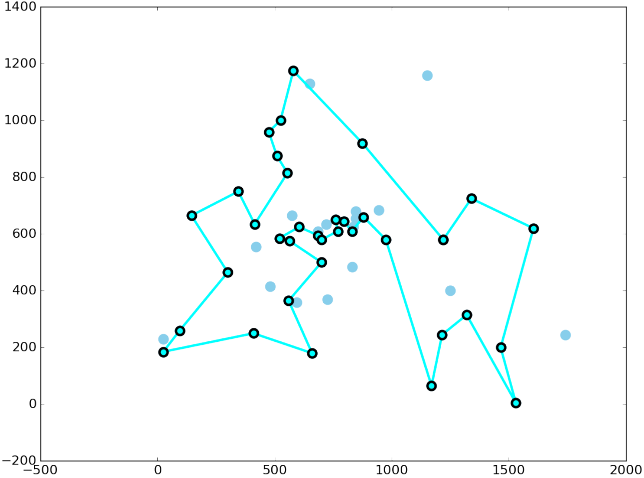
animation of the quick tour algorithm



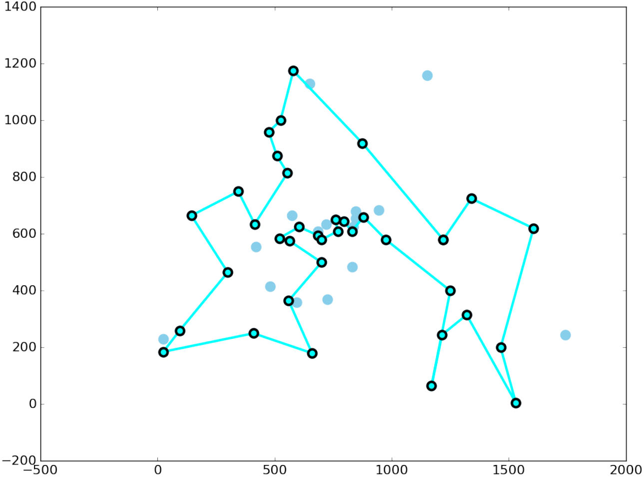
animation of the quick tour algorithm

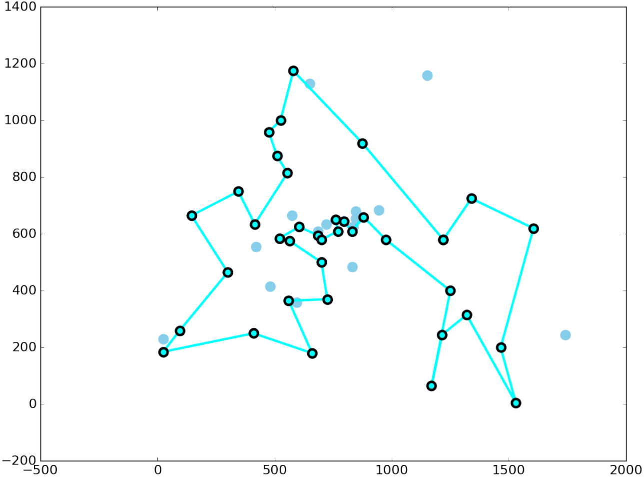


animation of the quick tour algorithm

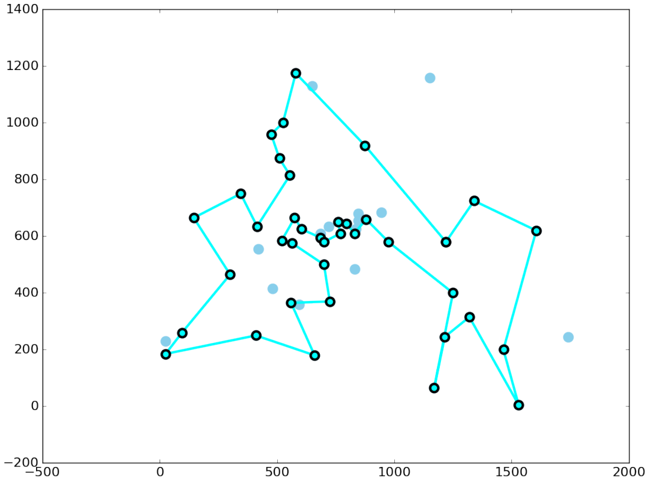


animation of the quick tour algorithm

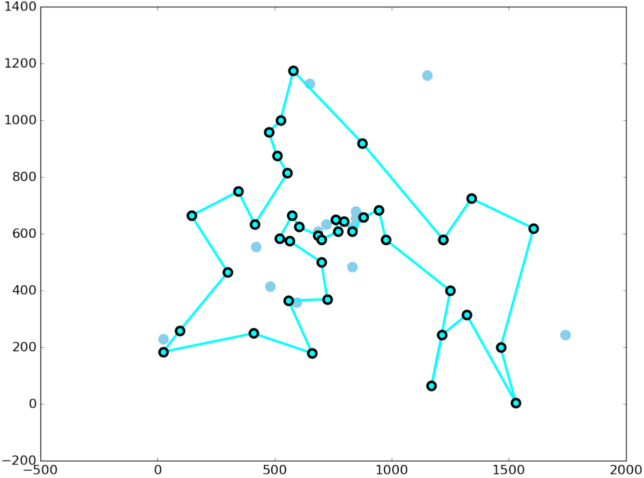



1400
1200
1000
800
600
400
200
0
−200
−500
0
500
1000
1500
2000

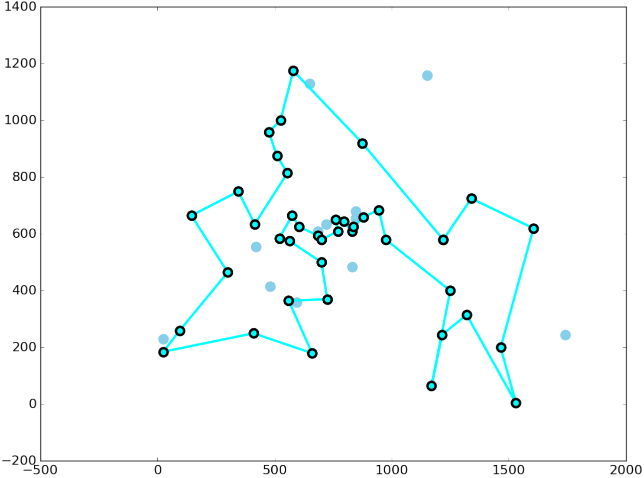
animation of the quick tour algorithm



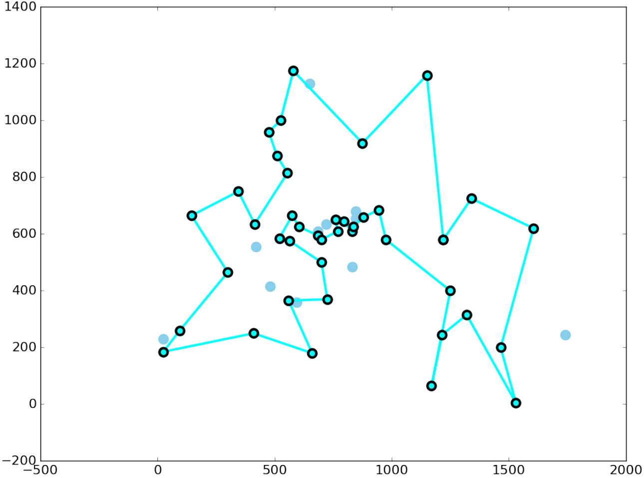
animation of the quick tour algorithm



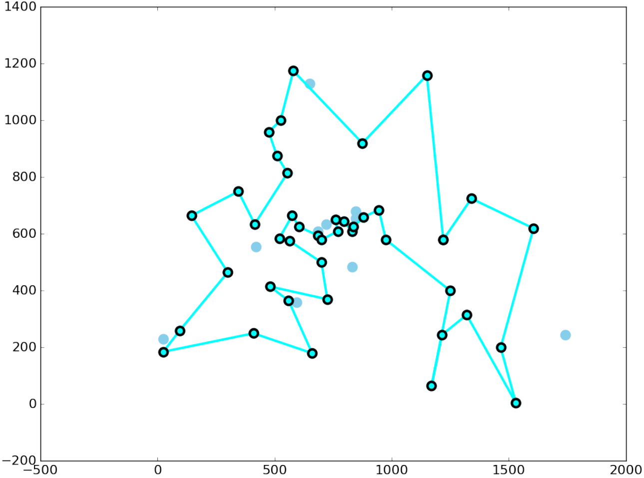
animation of the quick tour algorithm



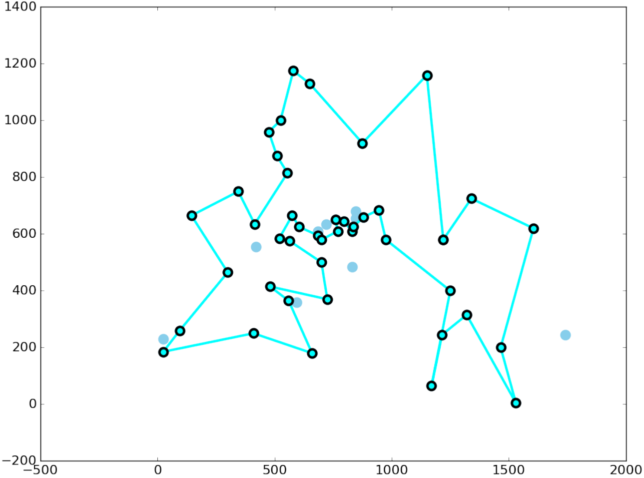
animation of the quick tour algorithm



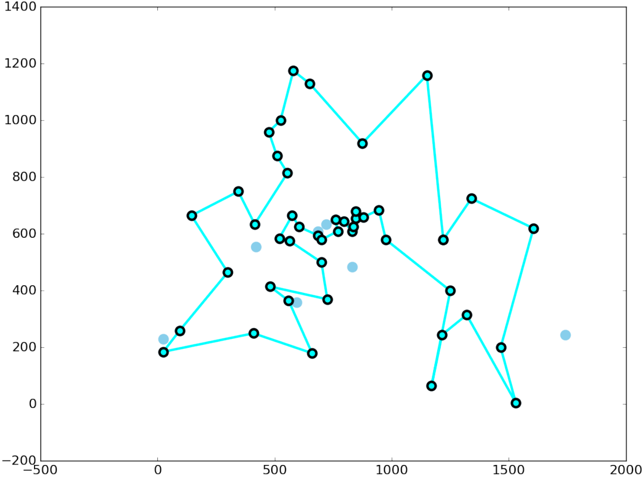
animation of the quick tour algorithm



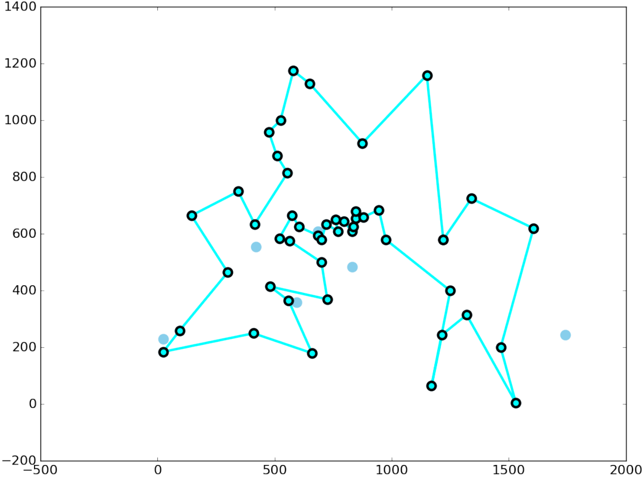
animation of the quick tour algorithm



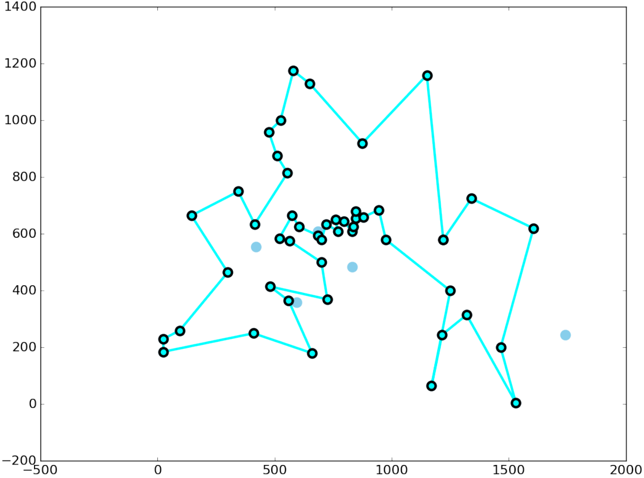
animation of the quick tour algorithm



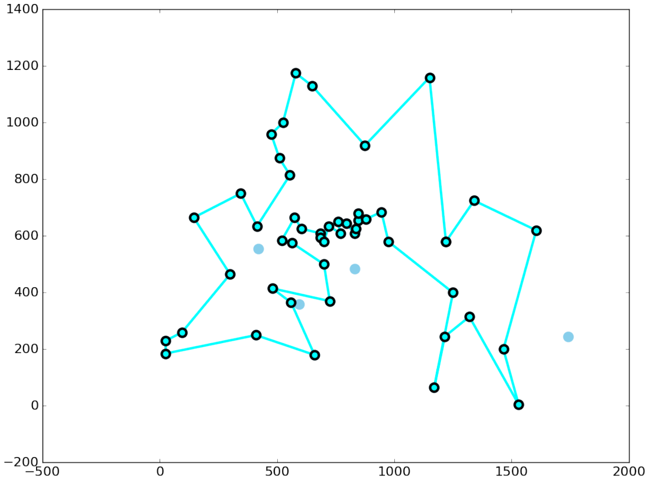
animation of the quick tour algorithm



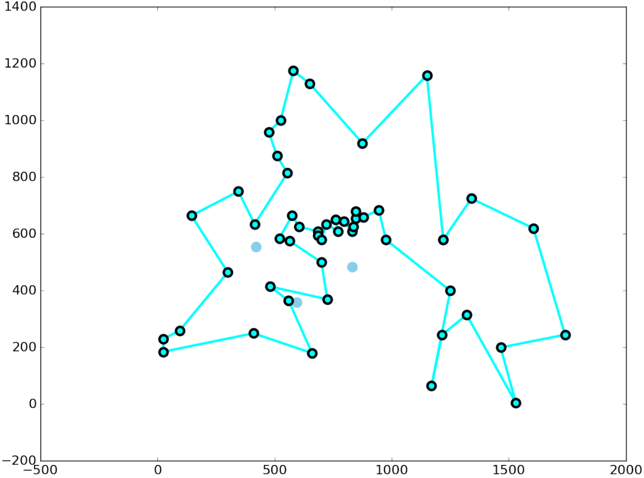
animation of the quick tour algorithm



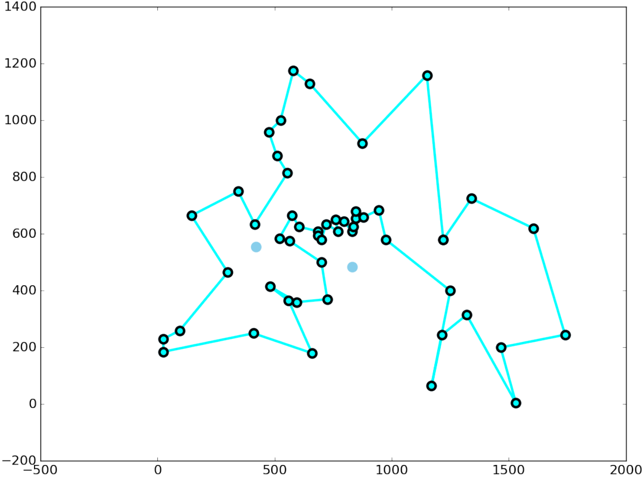
animation of the quick tour algorithm



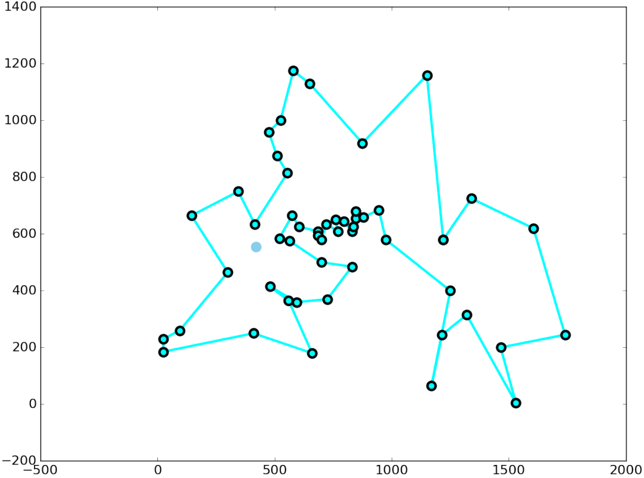
animation of the quick tour algorithm



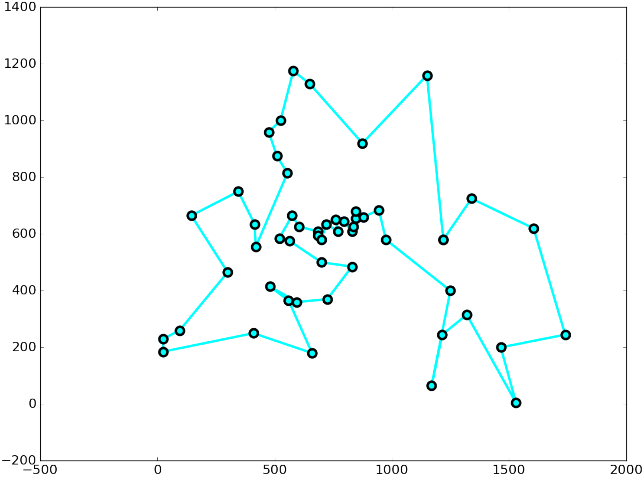
animation of the quick tour algorithm



animation of the quick tour algorithm



animation of the quick tour algorithm



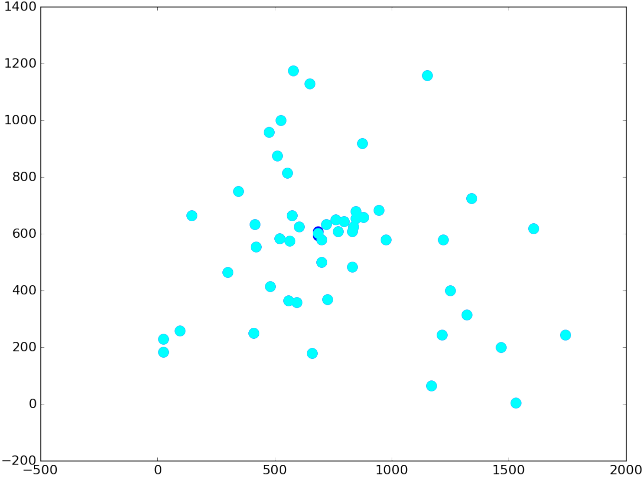
How does the pair-center algorithm work?

The pair-center tour algorithm is a contribution of my own for this course. It is a deterministic algorithm:

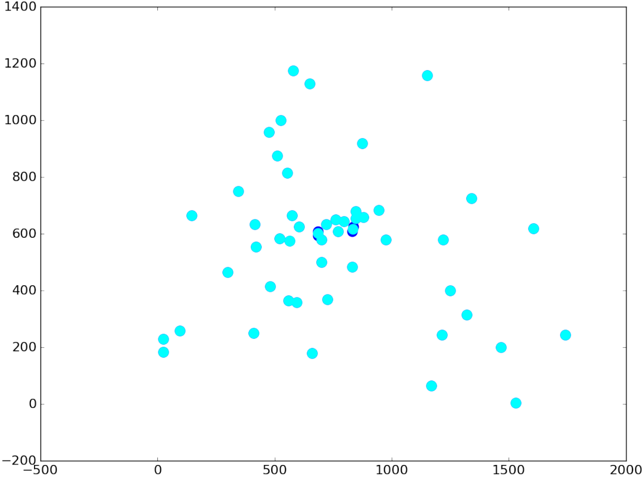
- with a bottom-up construction build a binary tree by replacing the/a closest pair of points by their center
- with a top-down construction build the tour by inserting the corresponding pairs in the best possible way

- the runtime is in $\mathcal{O}(n \log n)$, https://www.sciencedirect.com/science/article/pii/S1877750324002175
- implemented is $\mathcal{O}(n^3)$ (in Python) and a $\mathcal{O}(n \operatorname{polylog} n)$ version with more sophisticated data structures achieving quite low gaps

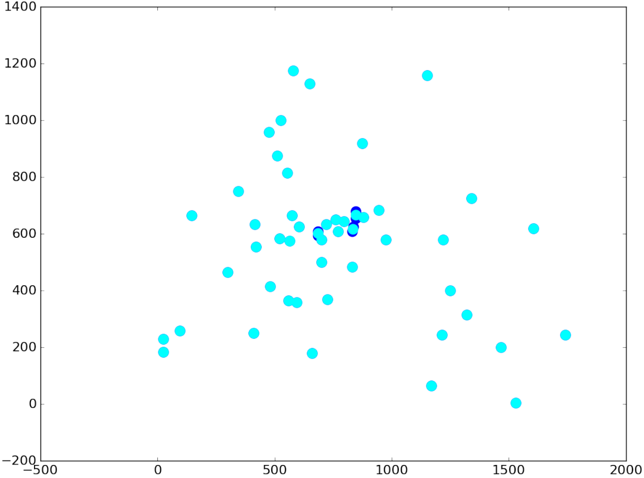
animation of the pair-center tour algorithm



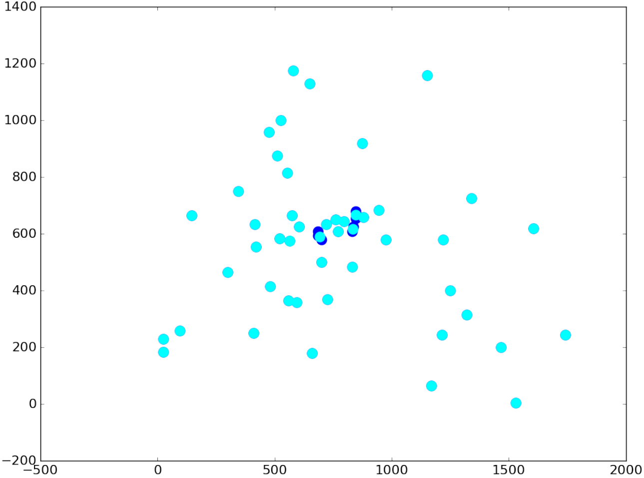
animation of the pair-center tour algorithm



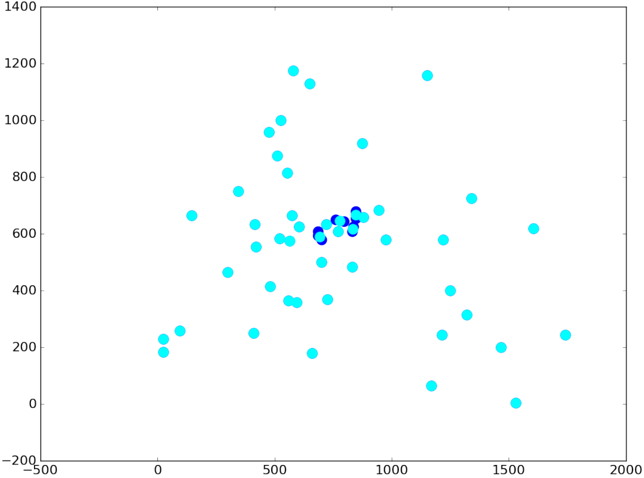
animation of the pair-center tour algorithm



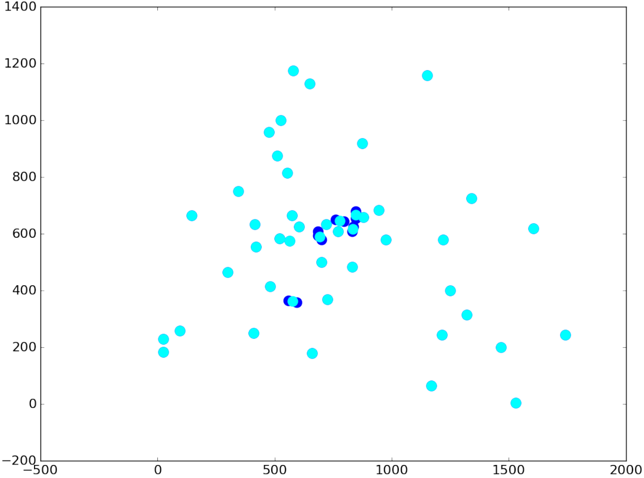
animation of the pair-center tour algorithm



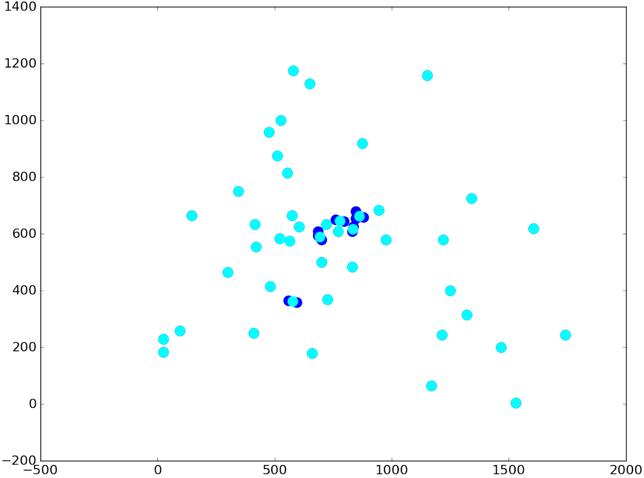
animation of the pair-center tour algorithm



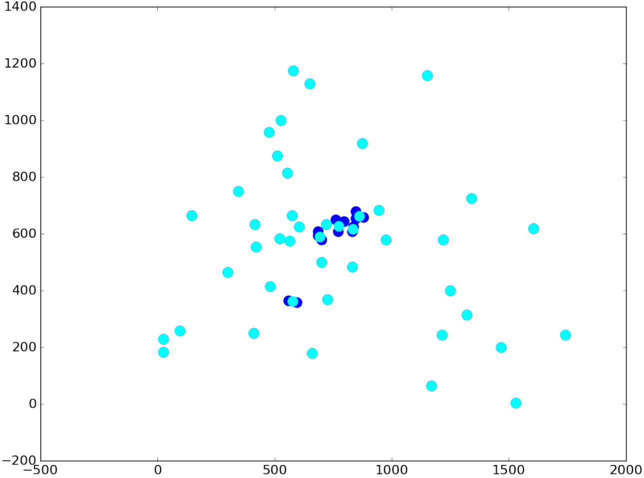
animation of the pair-center tour algorithm



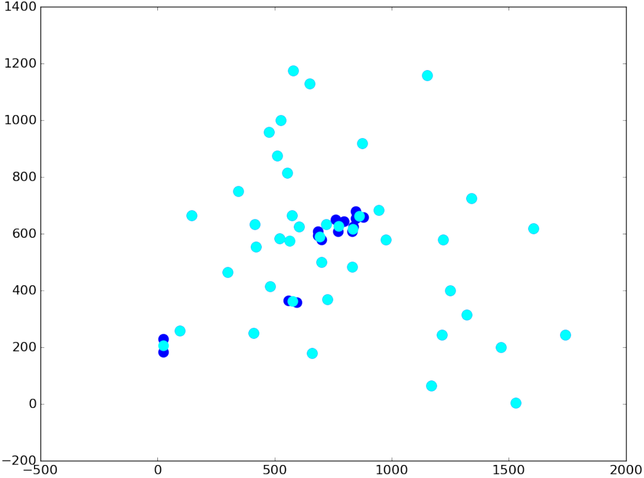
animation of the pair-center tour algorithm



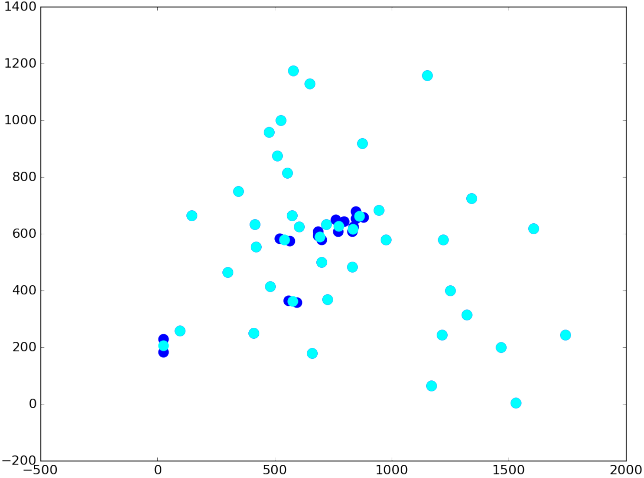
animation of the pair-center tour algorithm



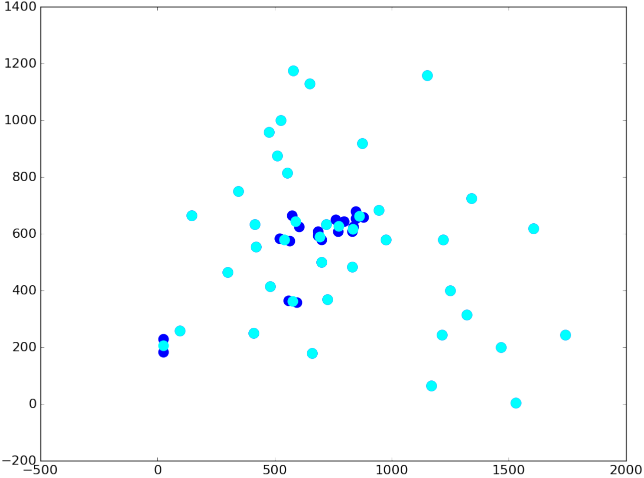
animation of the pair-center tour algorithm



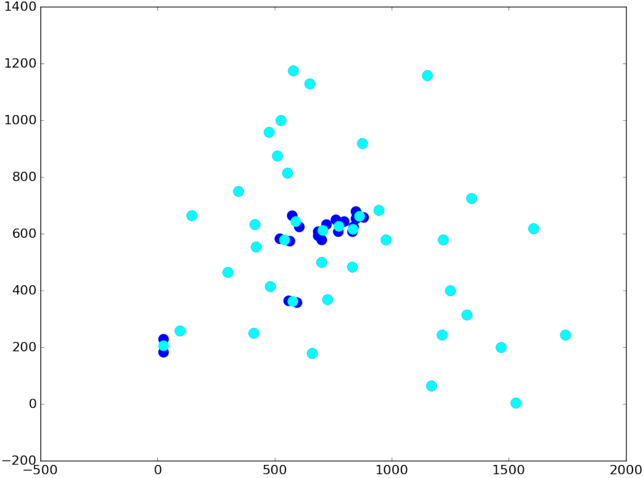
animation of the pair-center tour algorithm



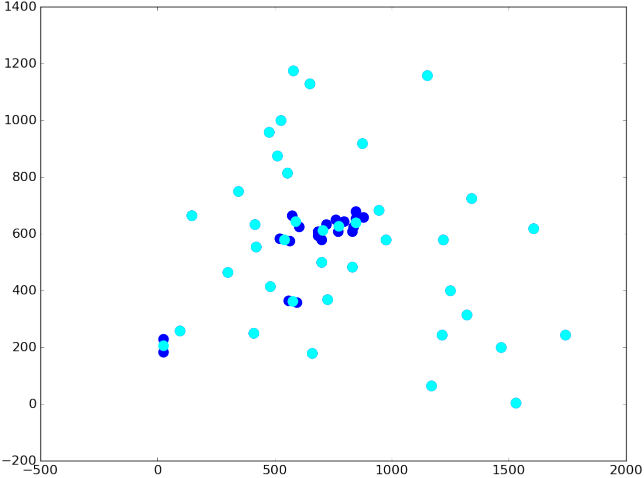
animation of the pair-center tour algorithm



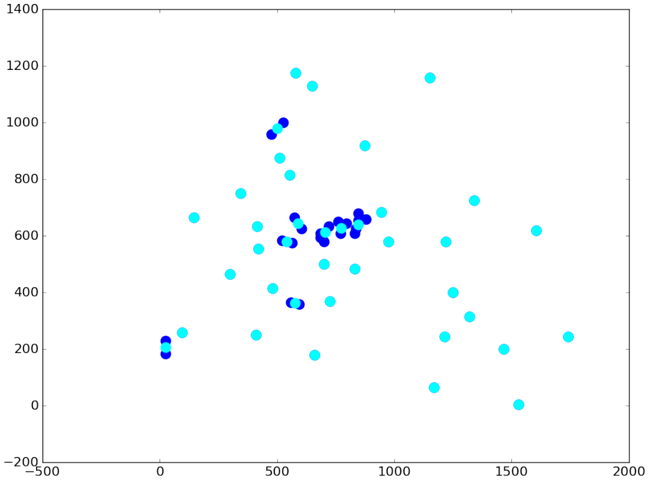
animation of the pair-center tour algorithm



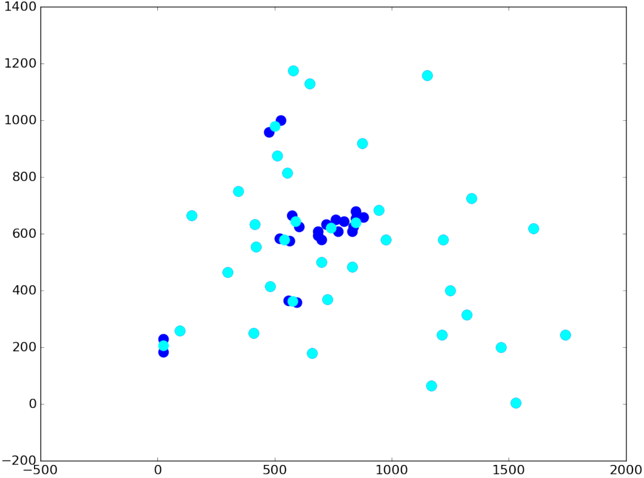
animation of the pair-center tour algorithm



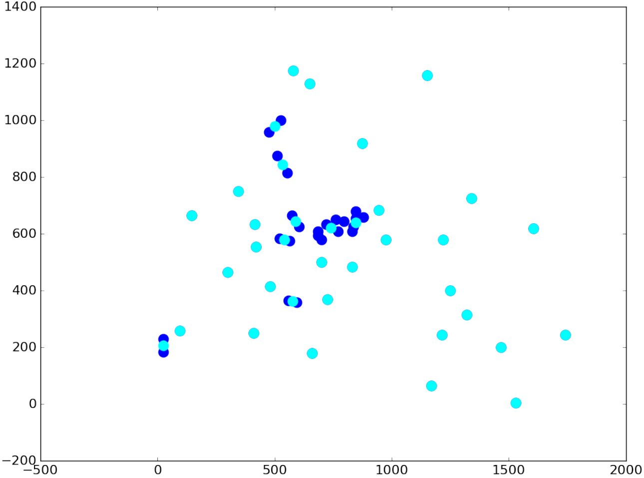
animation of the pair-center tour algorithm



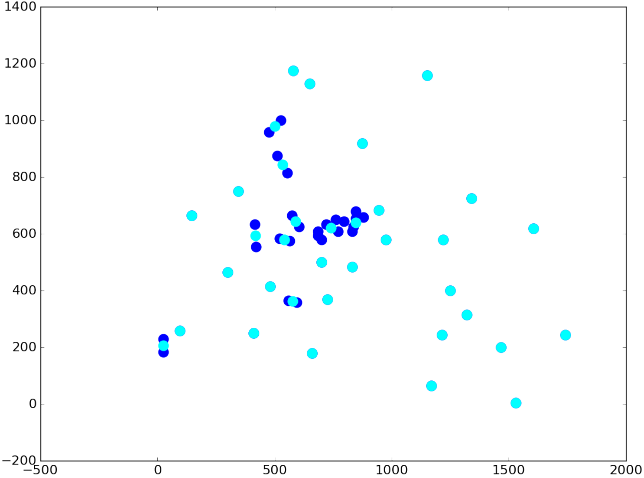
animation of the pair-center tour algorithm



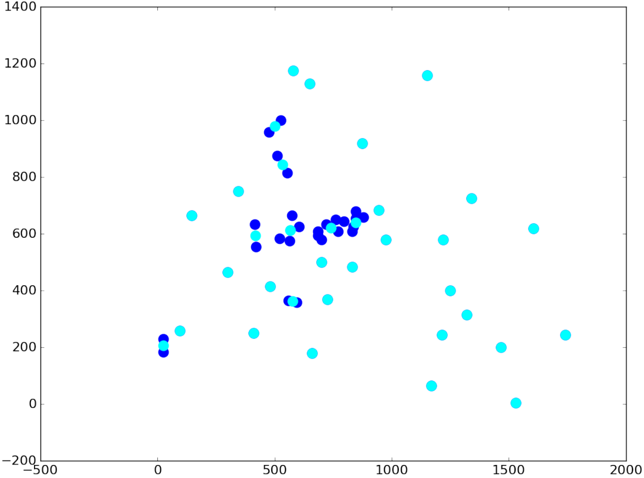
animation of the pair-center tour algorithm



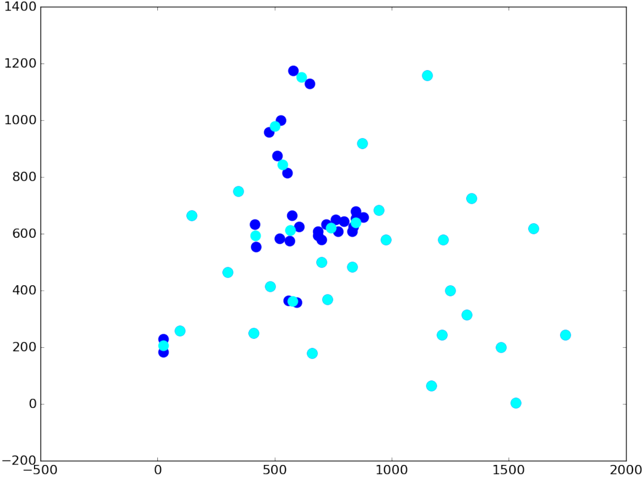
animation of the pair-center tour algorithm



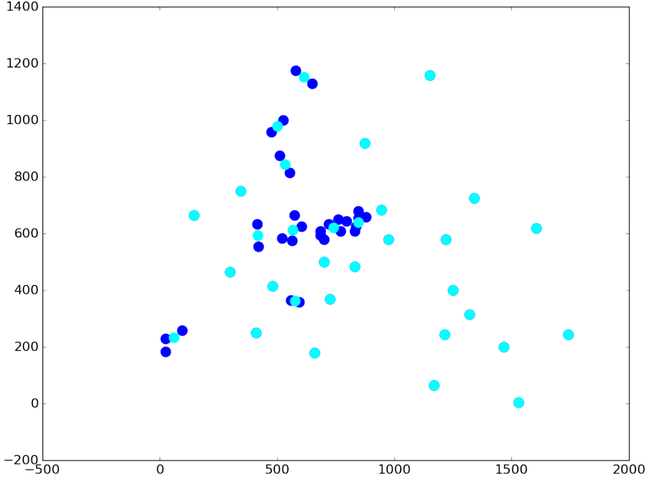
animation of the pair-center tour algorithm



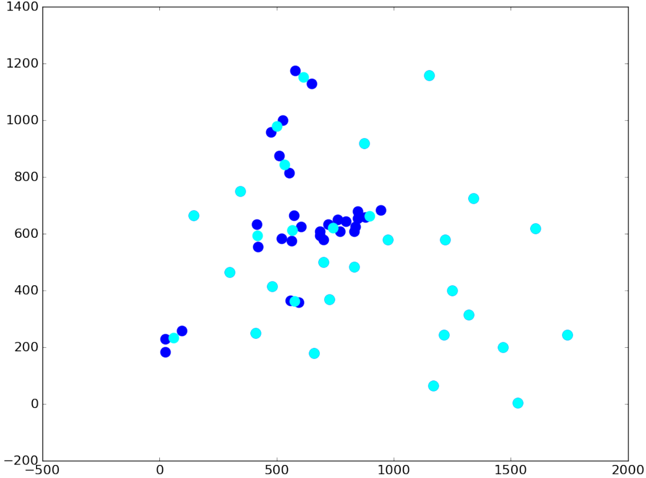
animation of the pair-center tour algorithm



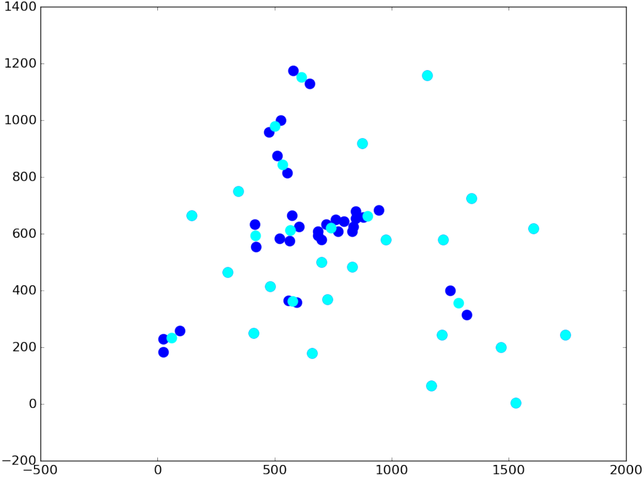
animation of the pair-center tour algorithm



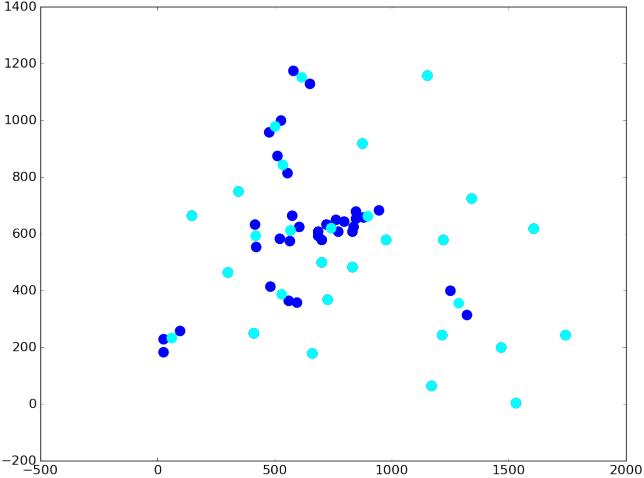
animation of the pair-center tour algorithm



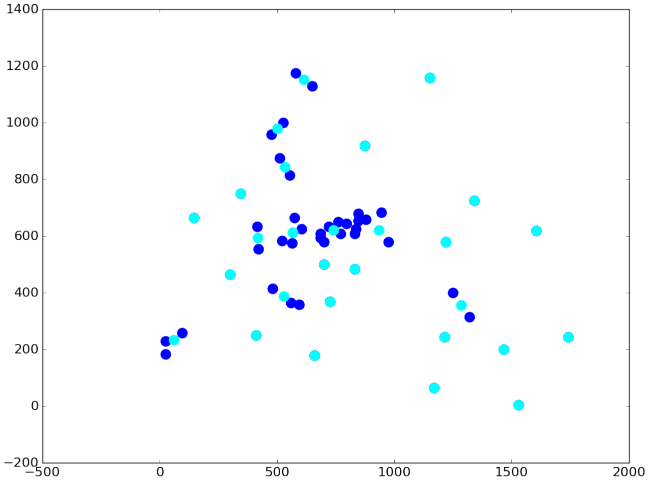
animation of the pair-center tour algorithm



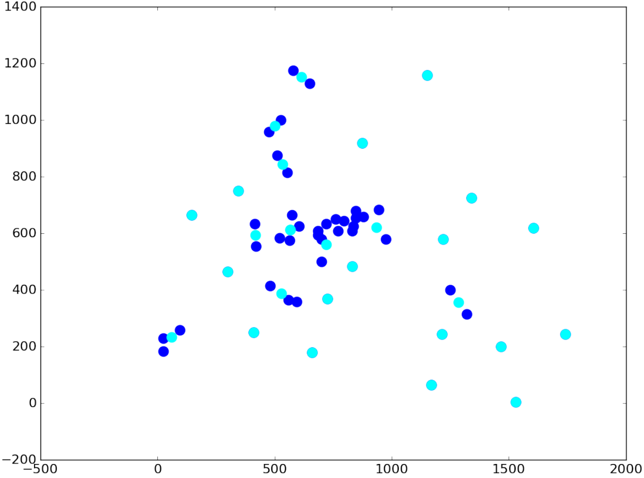
animation of the pair-center tour algorithm



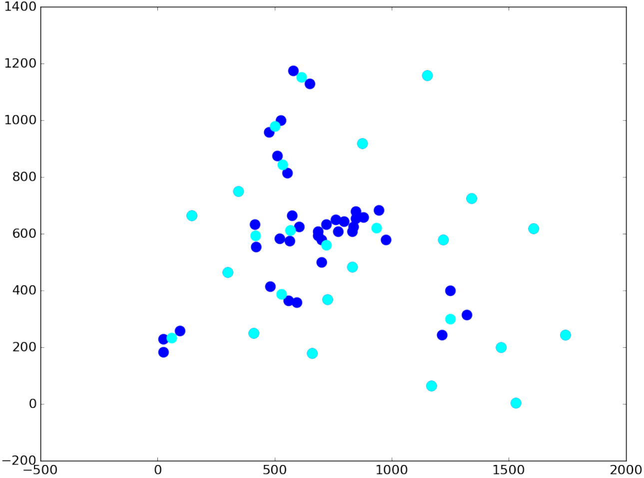
animation of the pair-center tour algorithm



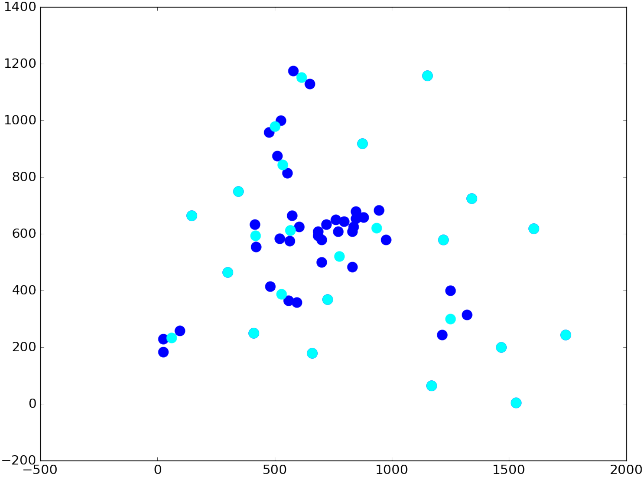
animation of the pair-center tour algorithm



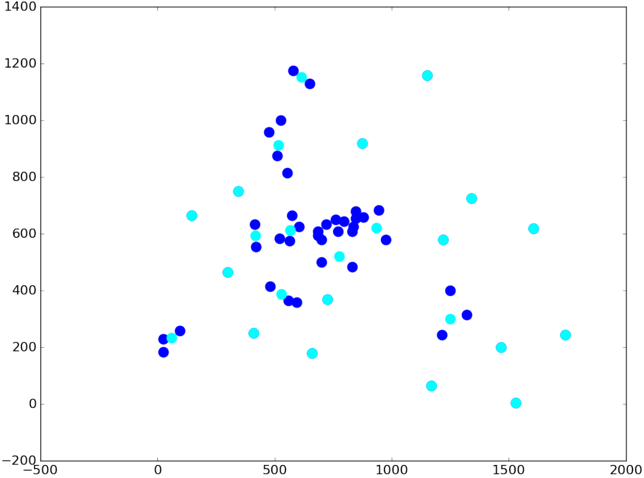
animation of the pair-center tour algorithm



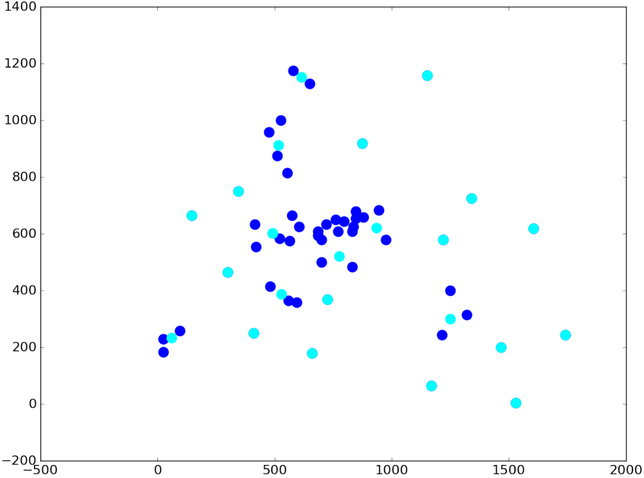
animation of the pair-center tour algorithm



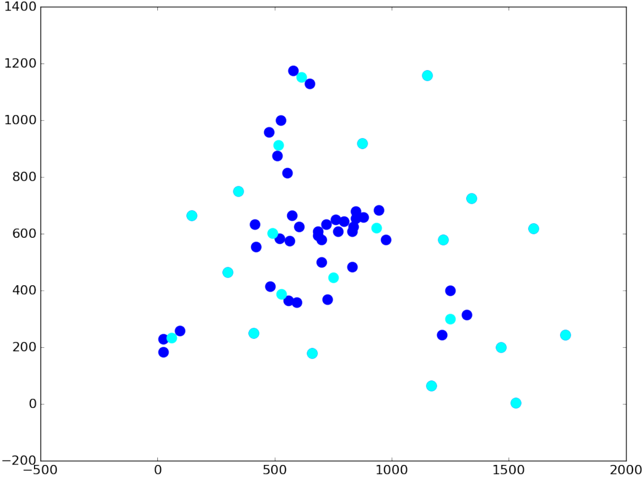
animation of the pair-center tour algorithm



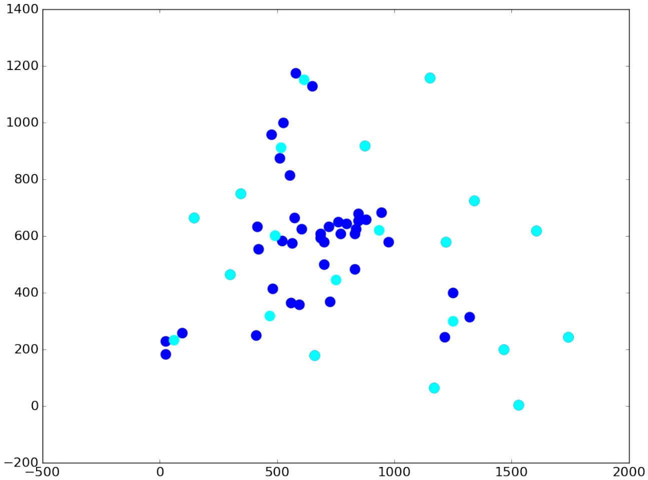
animation of the pair-center tour algorithm



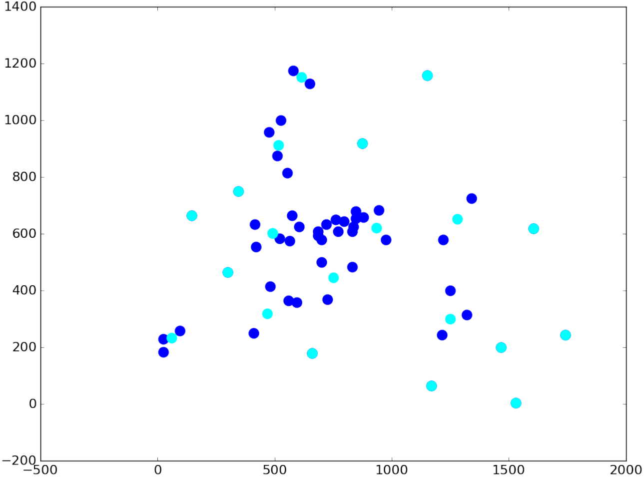
animation of the pair-center tour algorithm



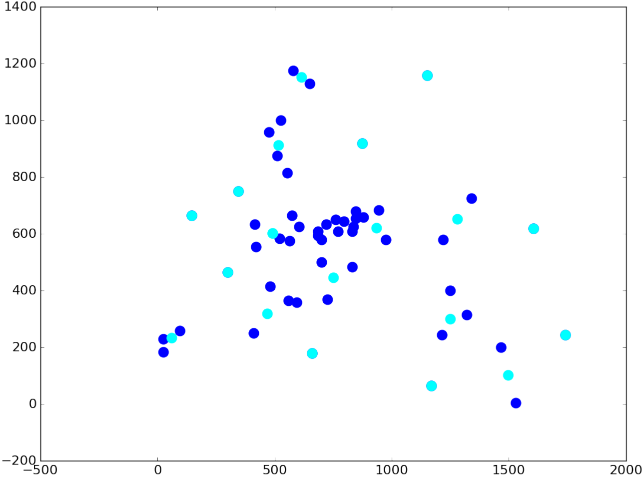
animation of the pair-center tour algorithm



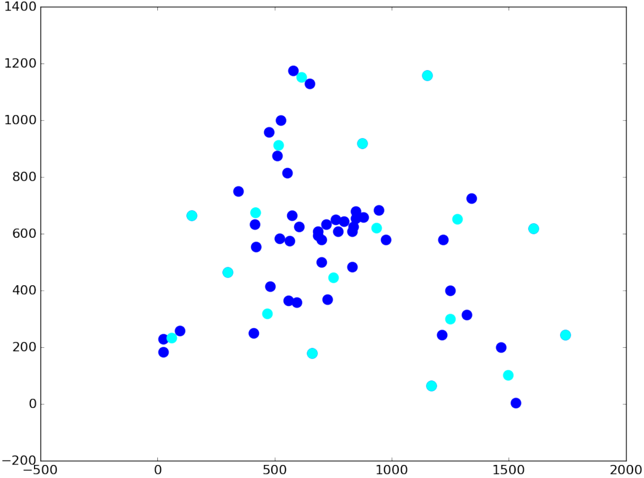
animation of the pair-center tour algorithm



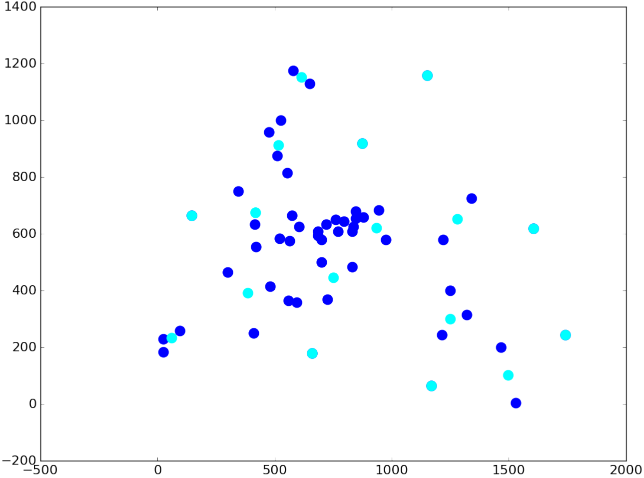
animation of the pair-center tour algorithm



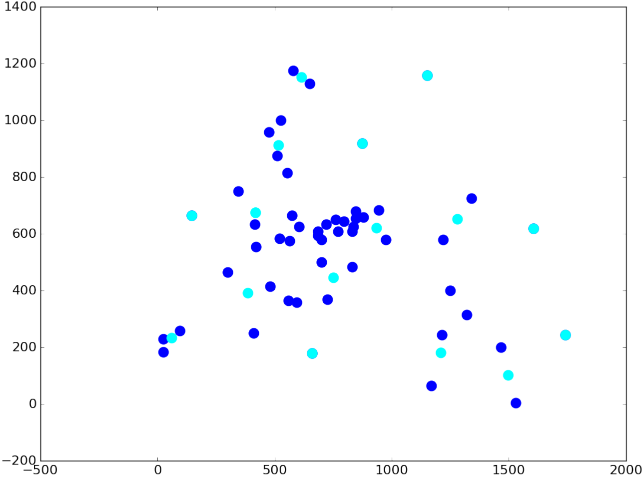
animation of the pair-center tour algorithm



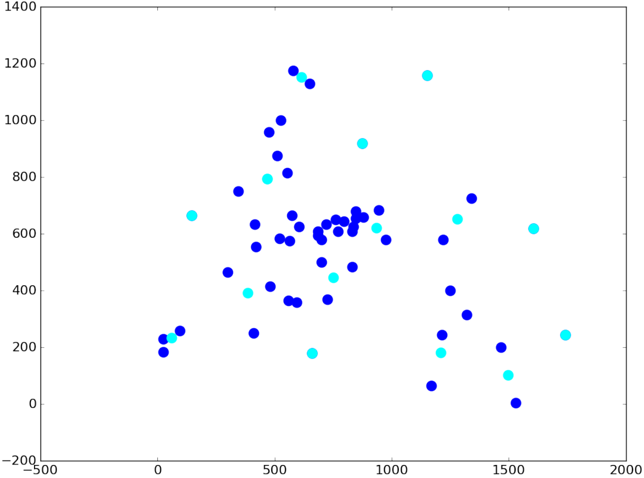
animation of the pair-center tour algorithm



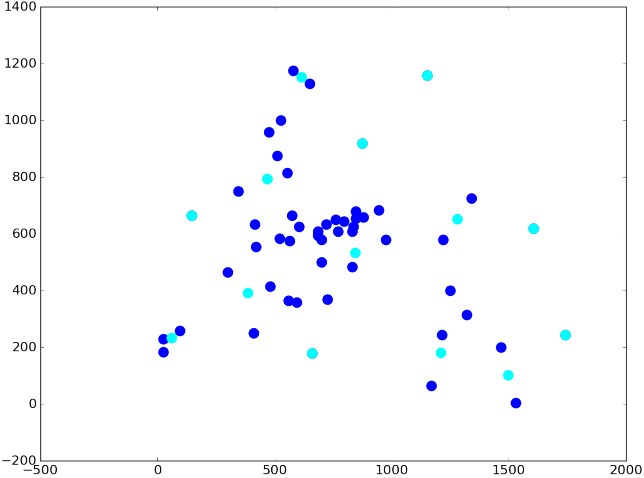
animation of the pair-center tour algorithm



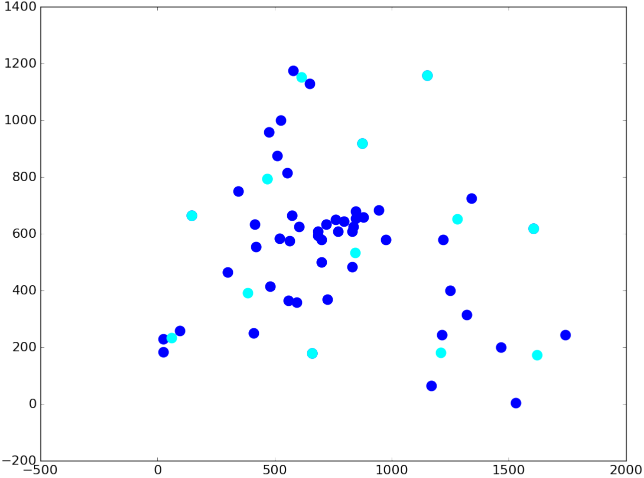
animation of the pair-center tour algorithm



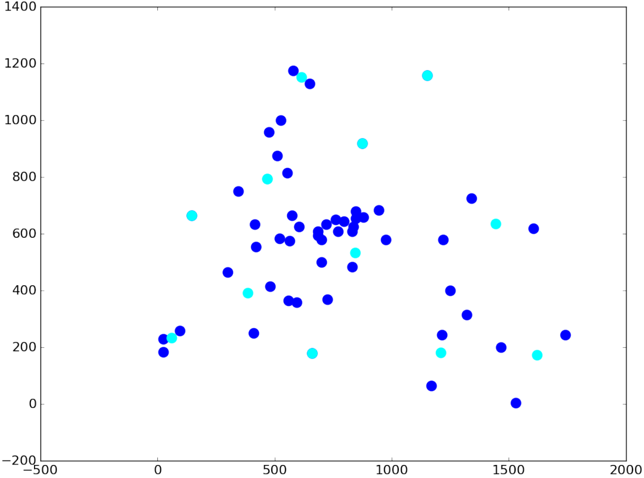
animation of the pair-center tour algorithm



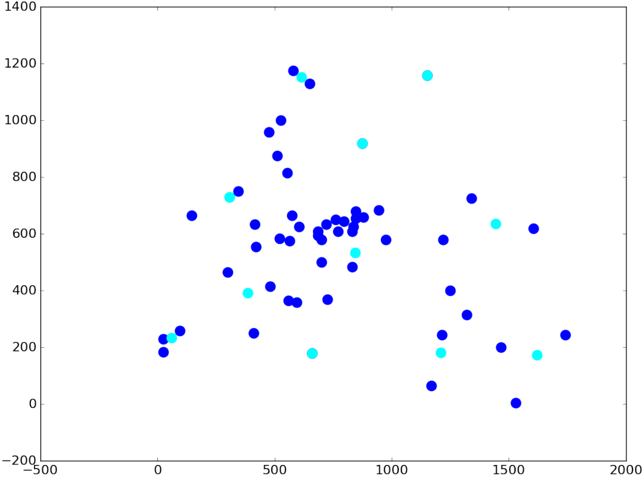
animation of the pair-center tour algorithm



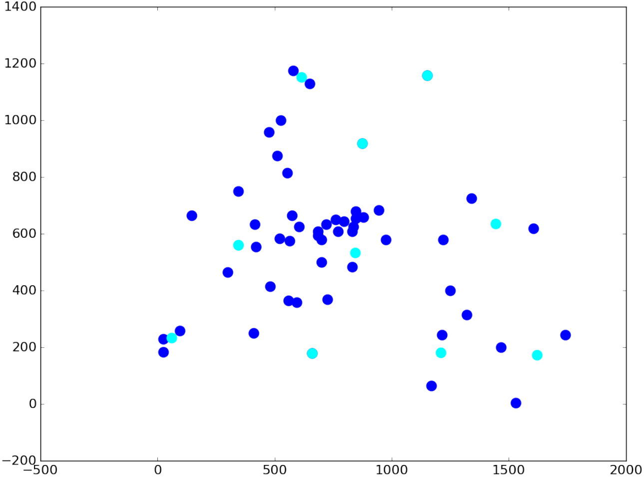
animation of the pair-center tour algorithm



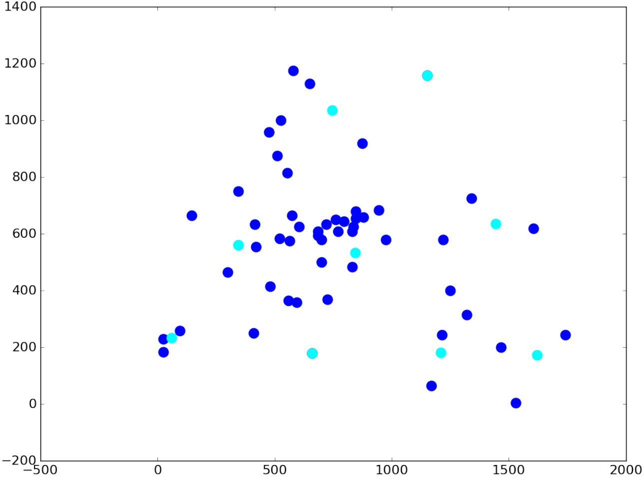
animation of the pair-center tour algorithm



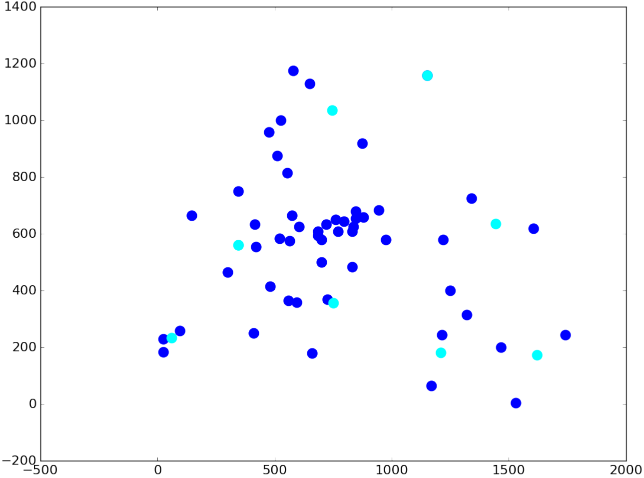
animation of the pair-center tour algorithm



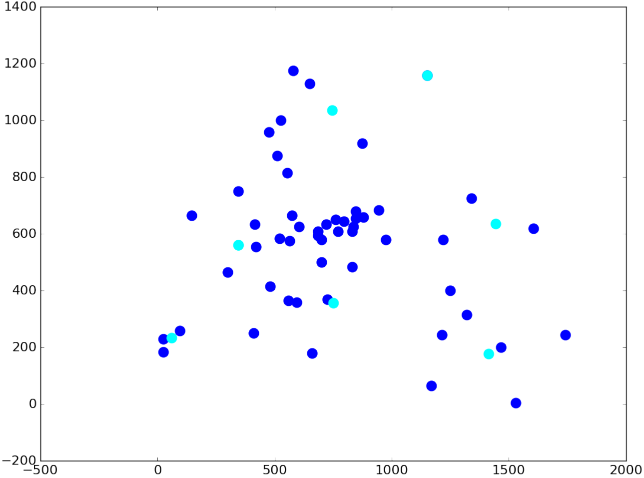
animation of the pair-center tour algorithm



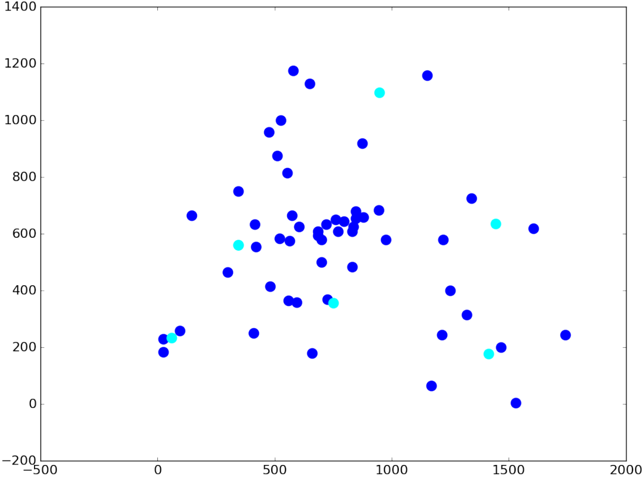
animation of the pair-center tour algorithm



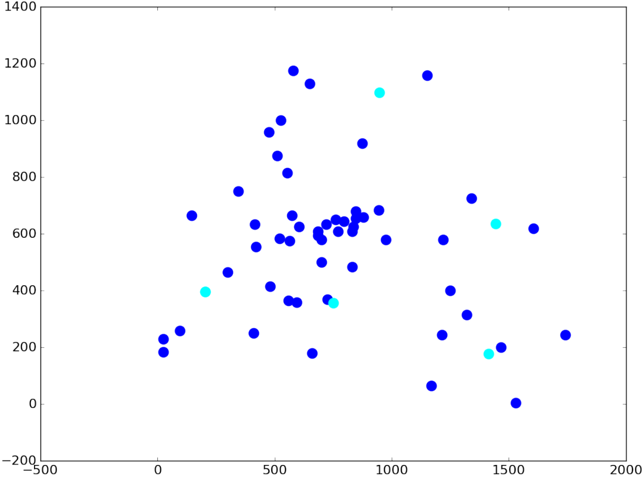
animation of the pair-center tour algorithm



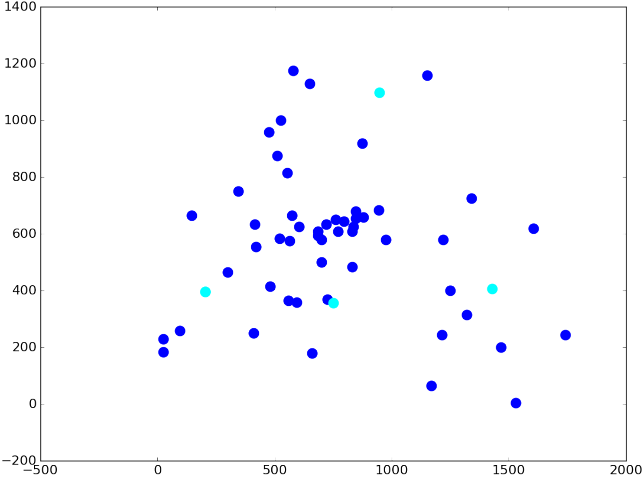
animation of the pair-center tour algorithm



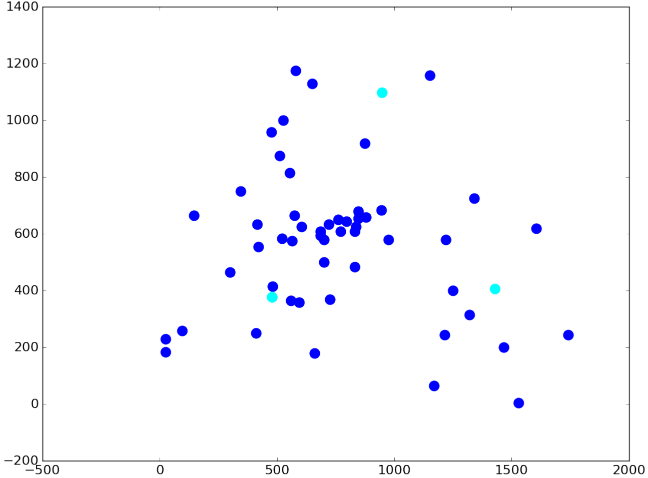
animation of the pair-center tour algorithm



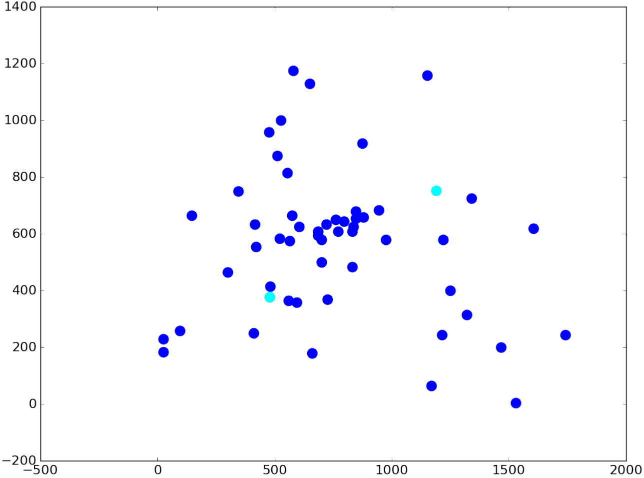
animation of the pair-center tour algorithm



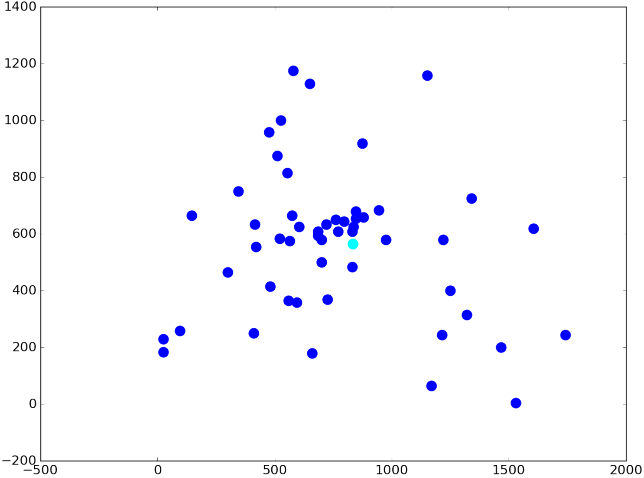
animation of the pair-center tour algorithm



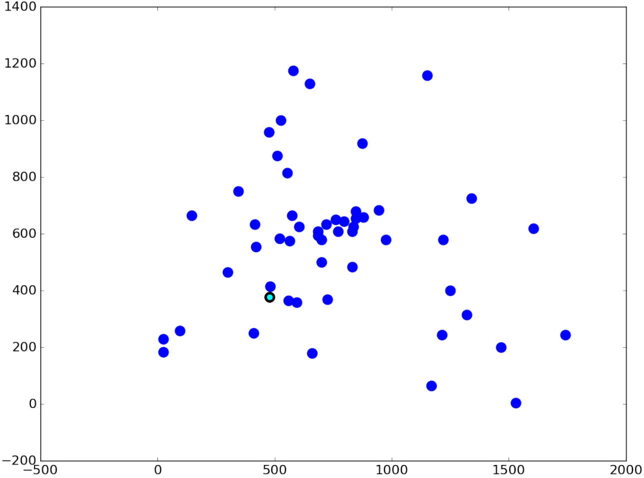
animation of the pair-center tour algorithm



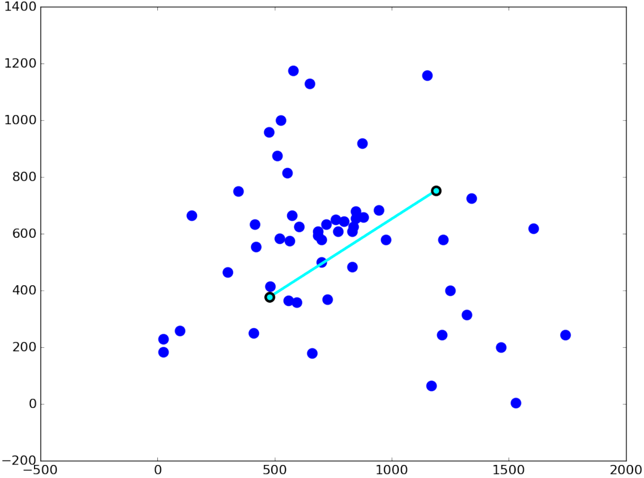
animation of the pair-center tour algorithm



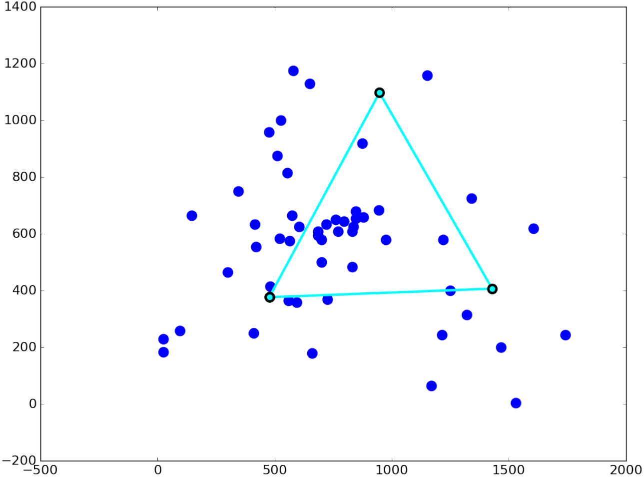
animation of the pair-center tour algorithm



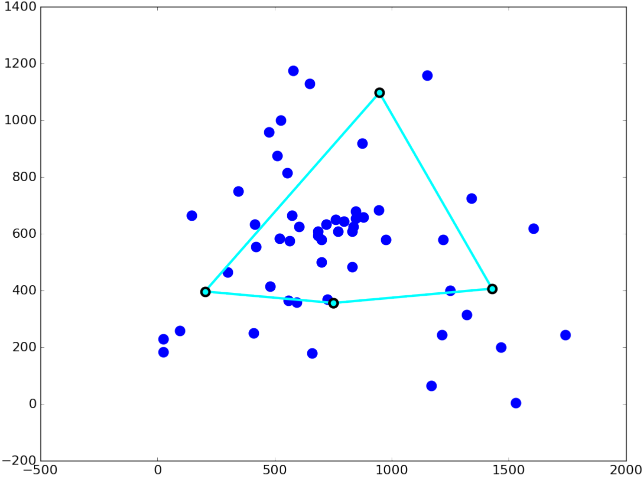
animation of the pair-center tour algorithm



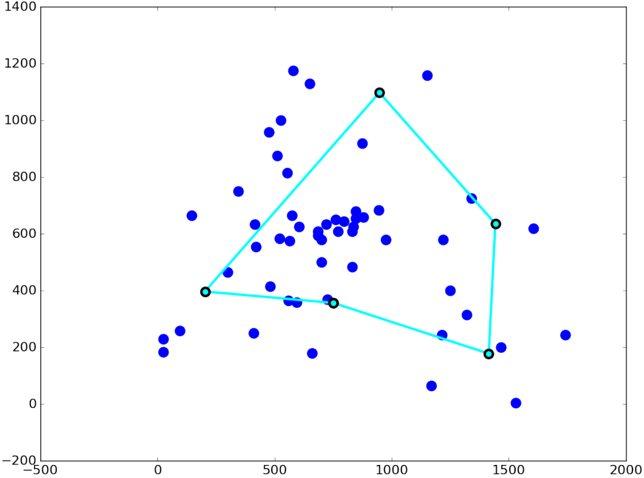
animation of the pair-center tour algorithm

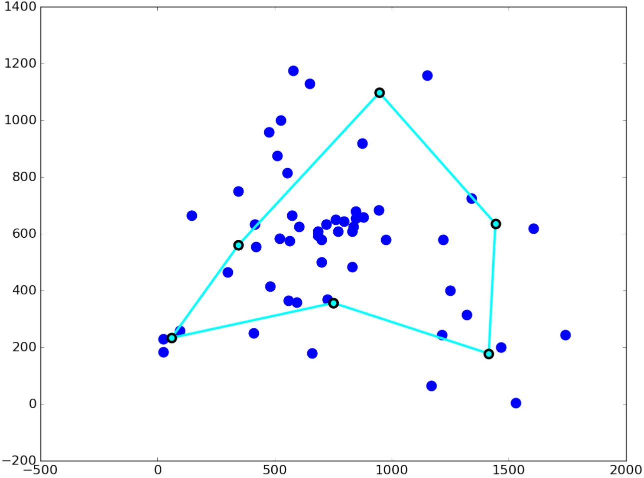


animation of the pair-center tour algorithm

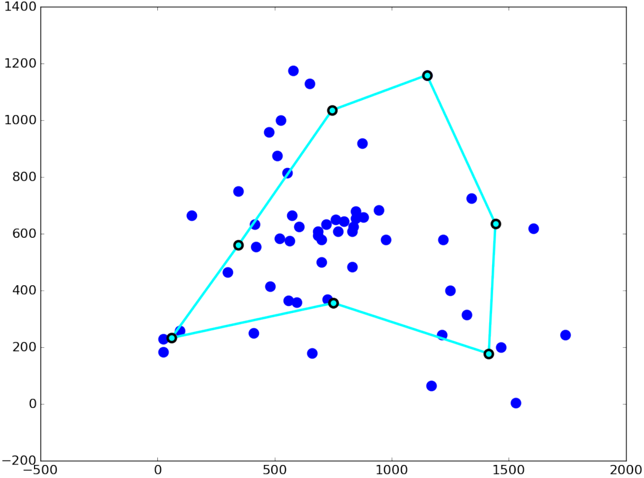


animation of the pair-center tour algorithm

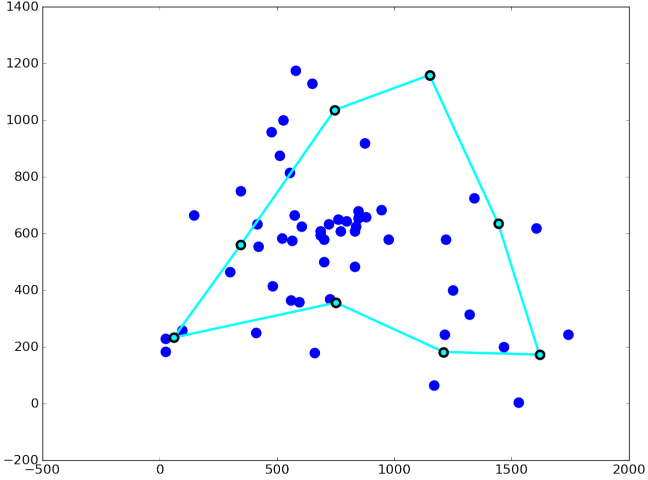



1400
1200
1000
800
600
400
200
0
−200
−500
0
500
1000
1500
2000

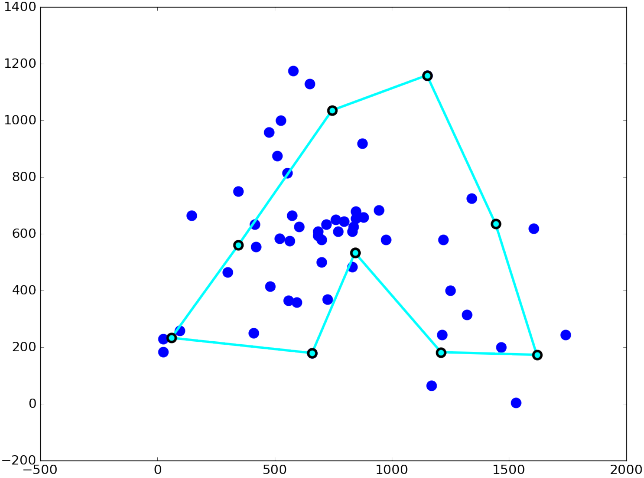
animation of the pair-center tour algorithm



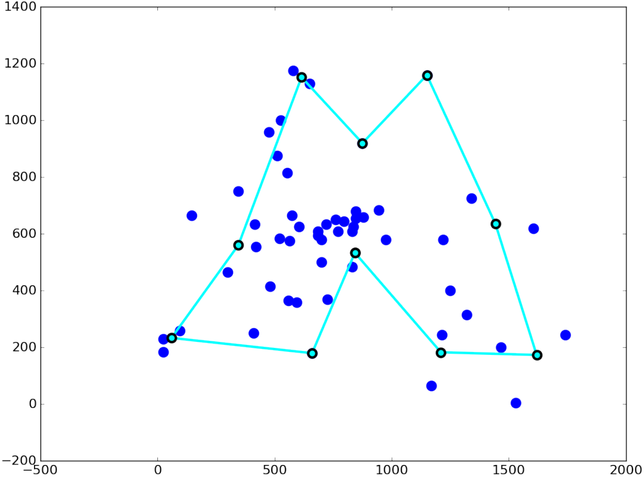
animation of the pair-center tour algorithm



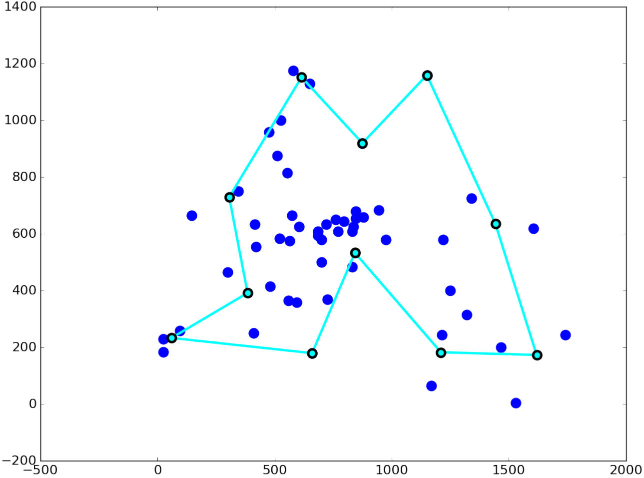
animation of the pair-center tour algorithm



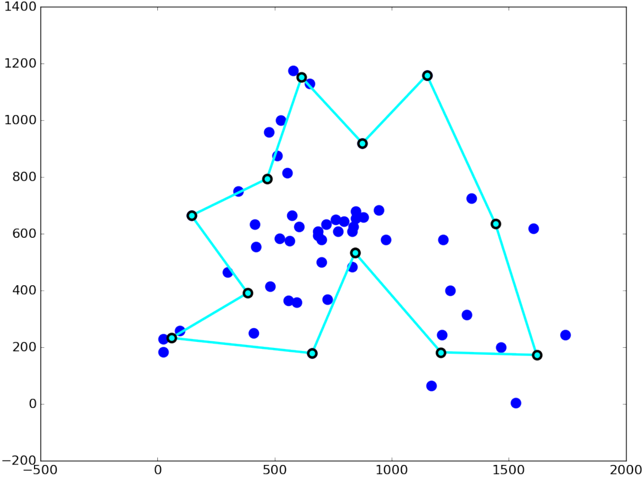
animation of the pair-center tour algorithm



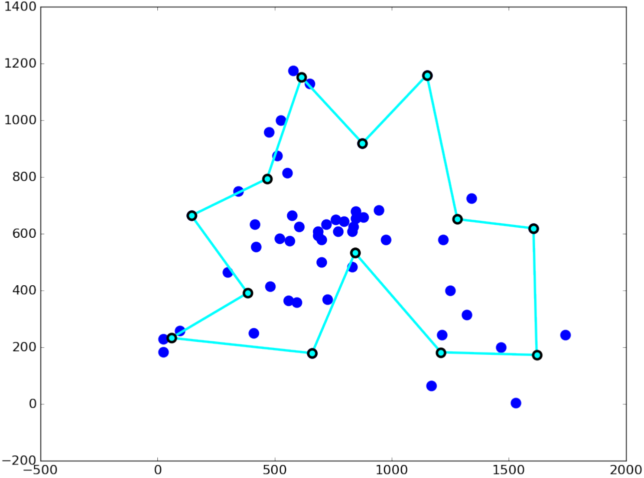
animation of the pair-center tour algorithm



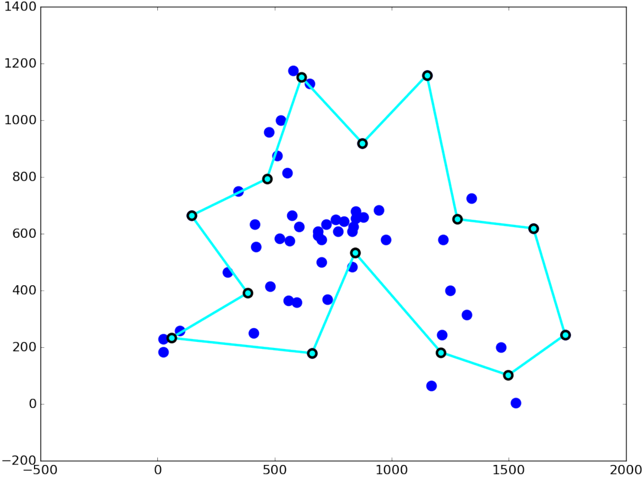
animation of the pair-center tour algorithm



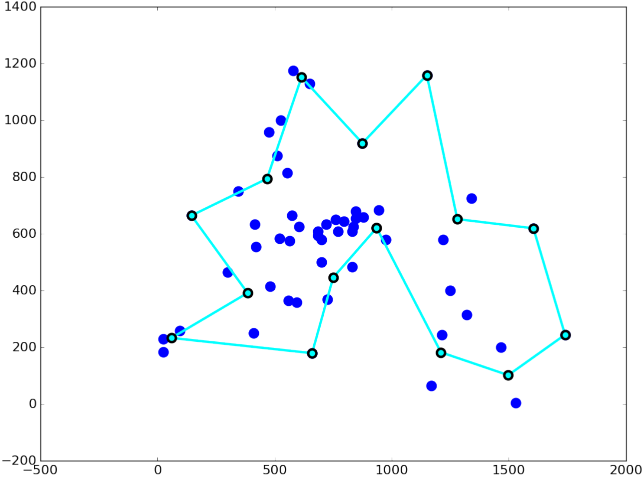
animation of the pair-center tour algorithm



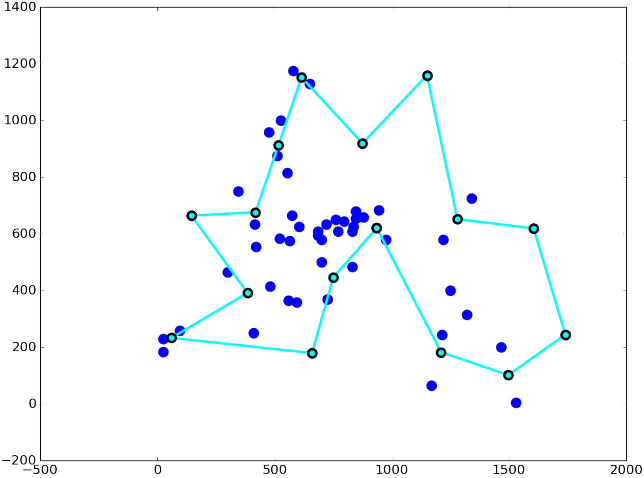
animation of the pair-center tour algorithm



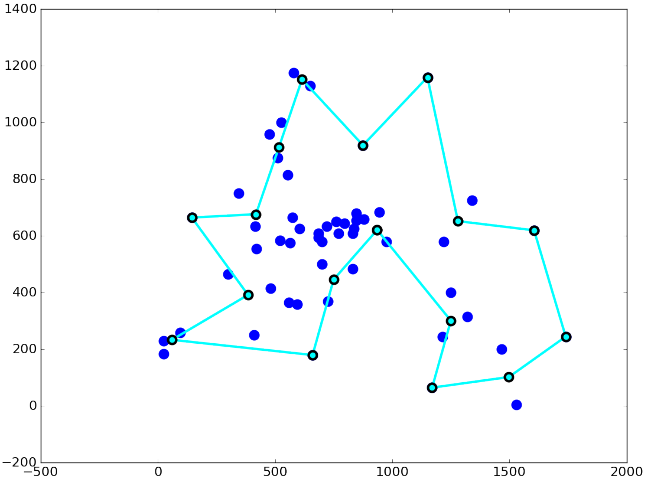
animation of the pair-center tour algorithm



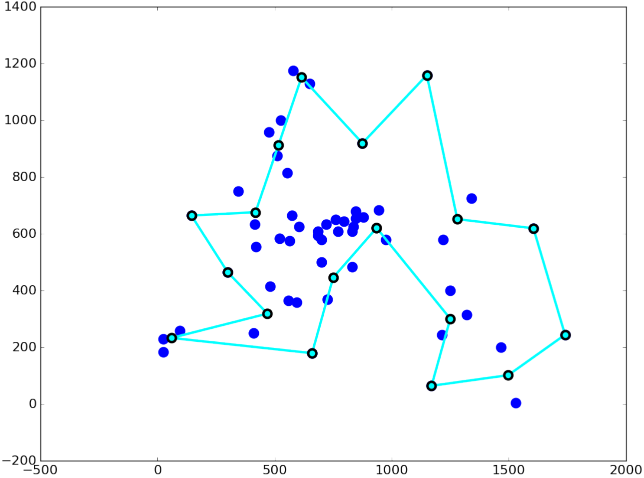
animation of the pair-center tour algorithm



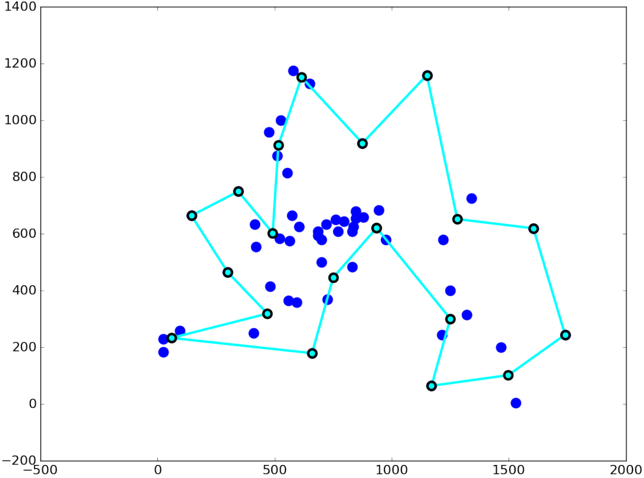
animation of the pair-center tour algorithm



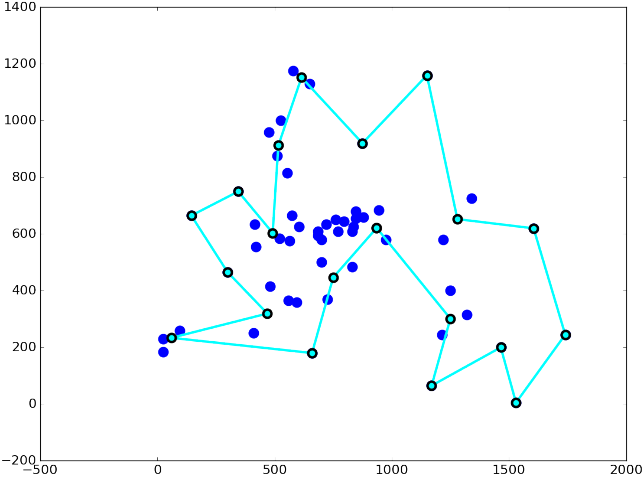
animation of the pair-center tour algorithm



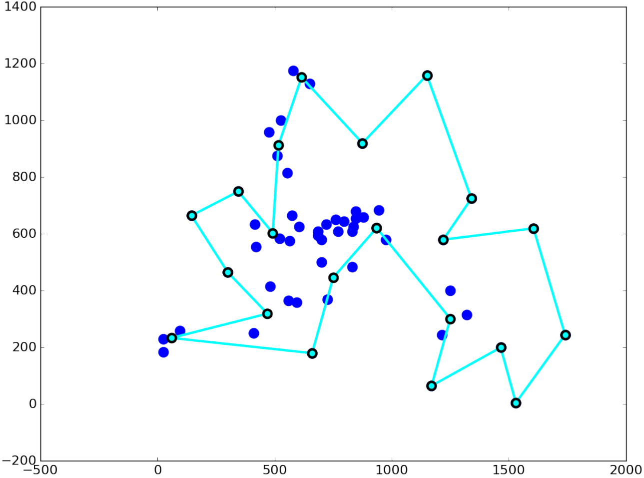
animation of the pair-center tour algorithm



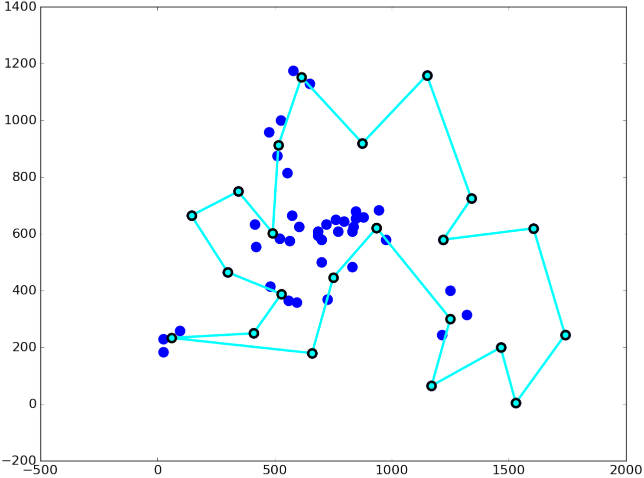
animation of the pair-center tour algorithm



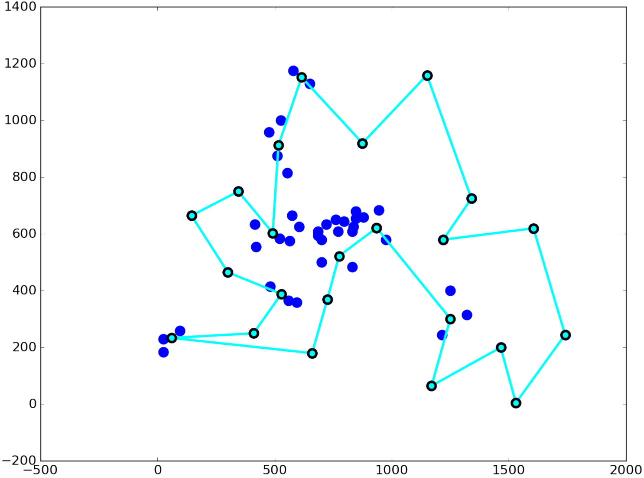
animation of the pair-center tour algorithm



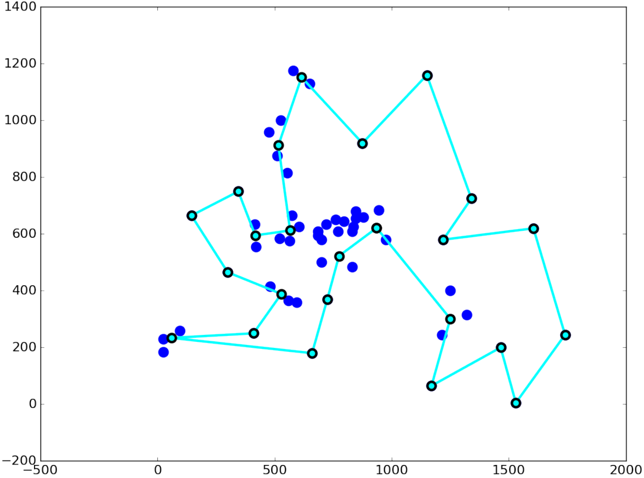
animation of the pair-center tour algorithm



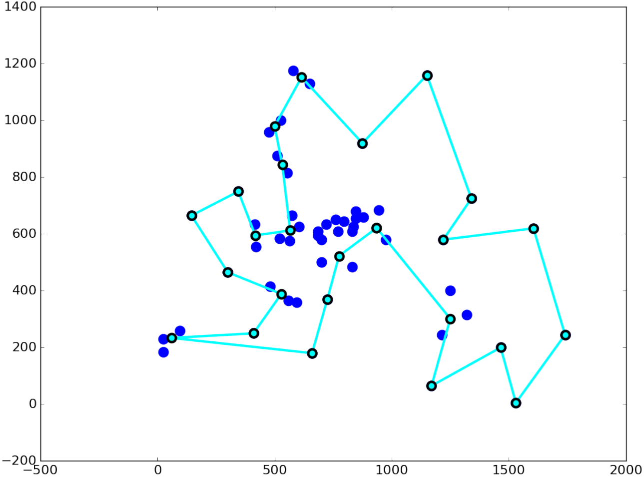
animation of the pair-center tour algorithm



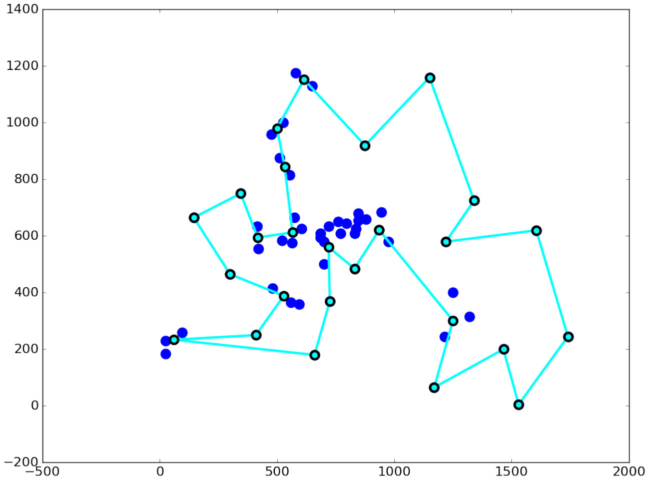
animation of the pair-center tour algorithm



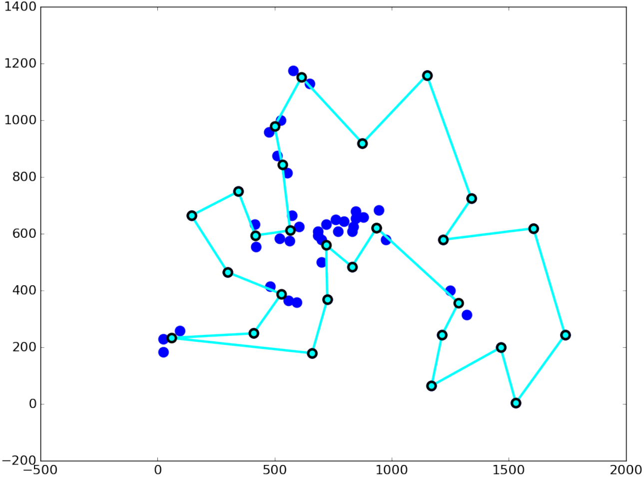
animation of the pair-center tour algorithm



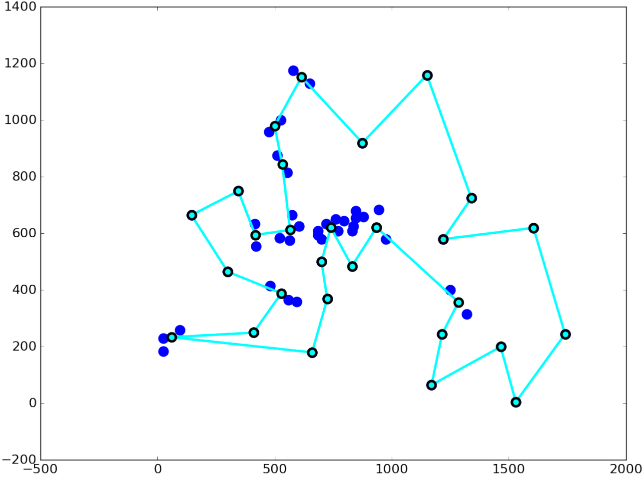
animation of the pair-center tour algorithm



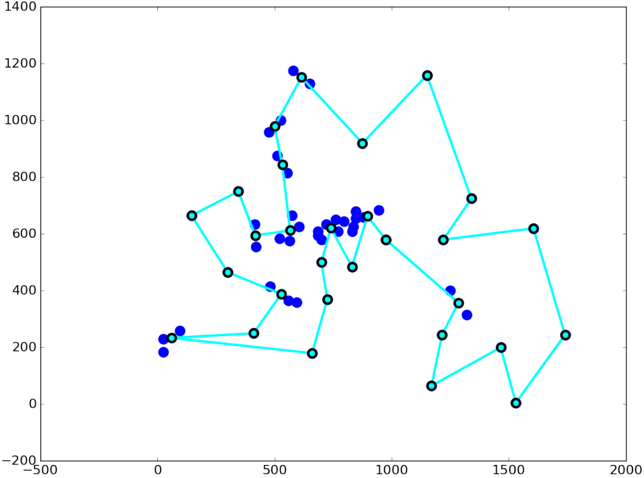
animation of the pair-center tour algorithm



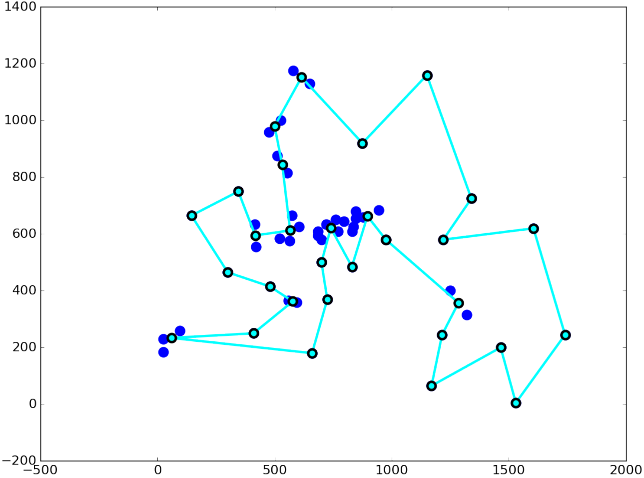
animation of the pair-center tour algorithm



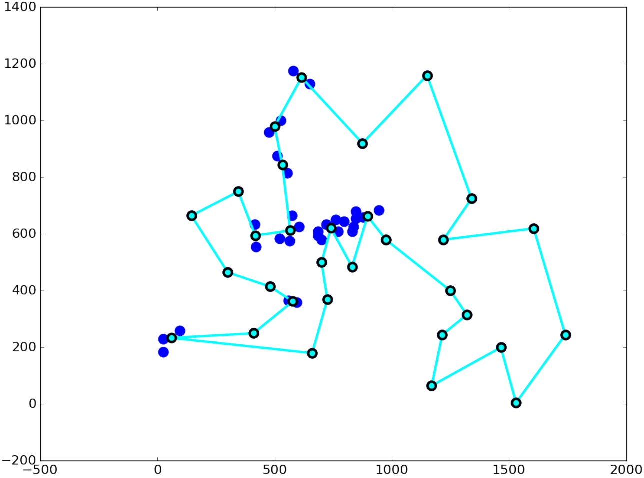
animation of the pair-center tour algorithm



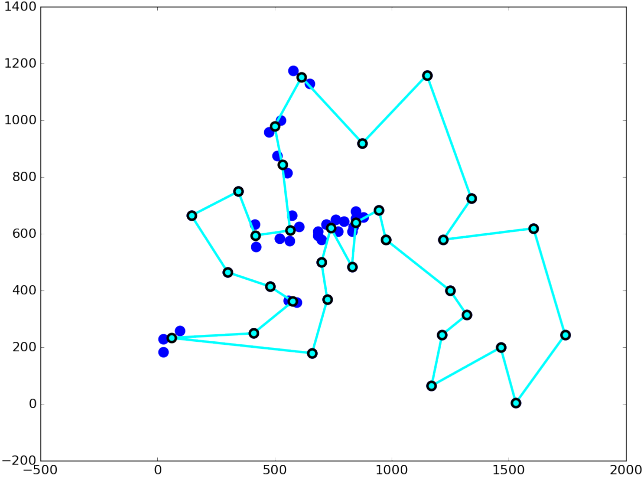
animation of the pair-center tour algorithm



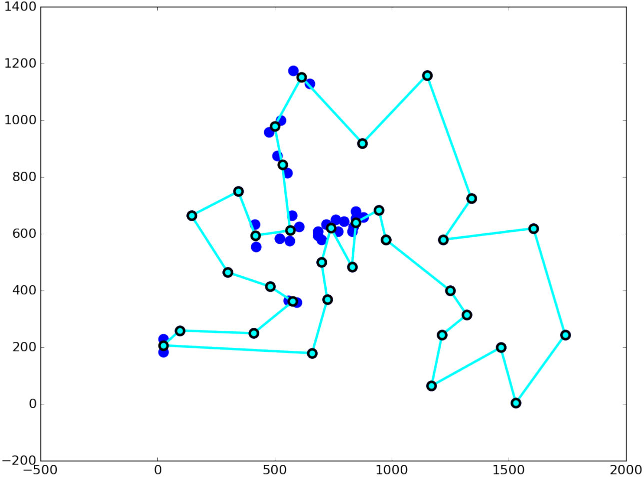
animation of the pair-center tour algorithm



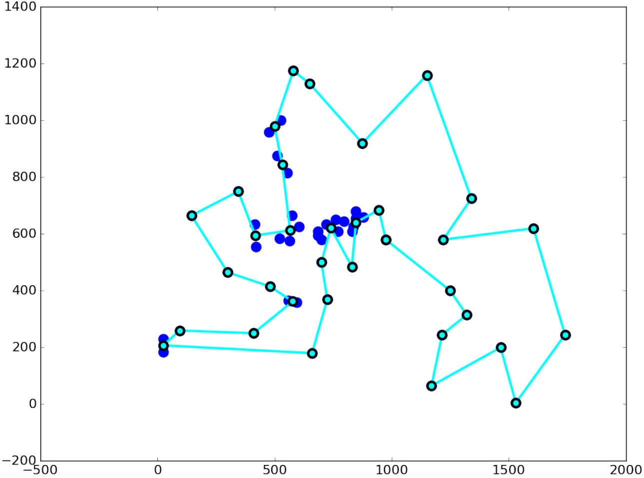
animation of the pair-center tour algorithm



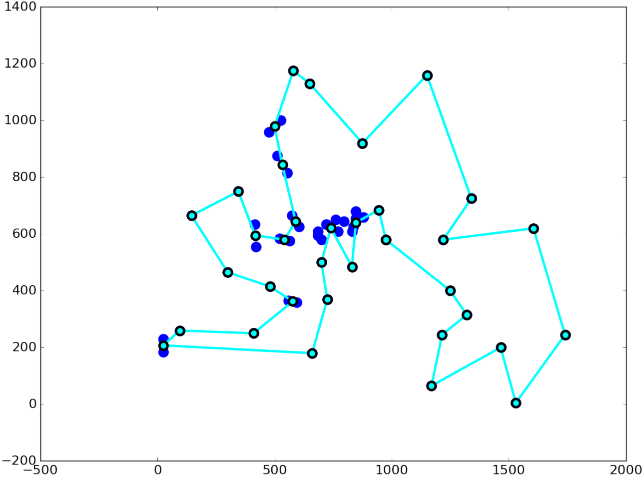
animation of the pair-center tour algorithm



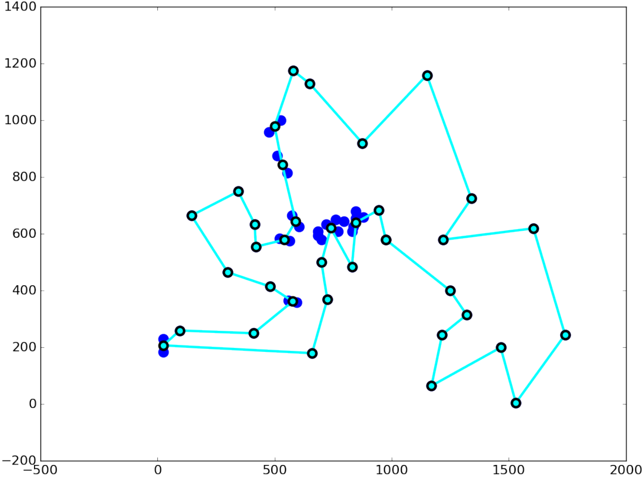
animation of the pair-center tour algorithm



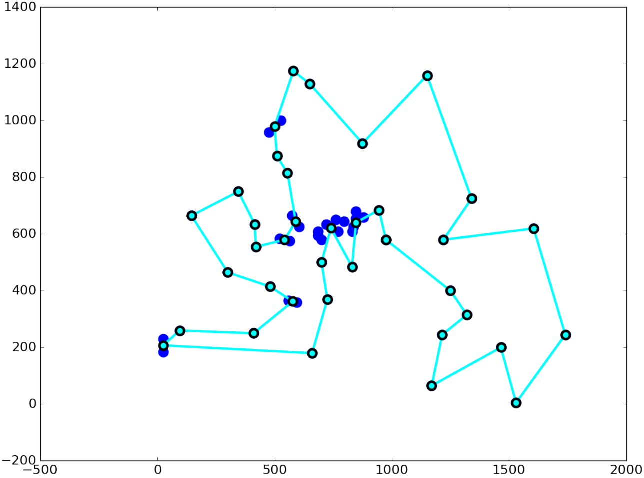
animation of the pair-center tour algorithm



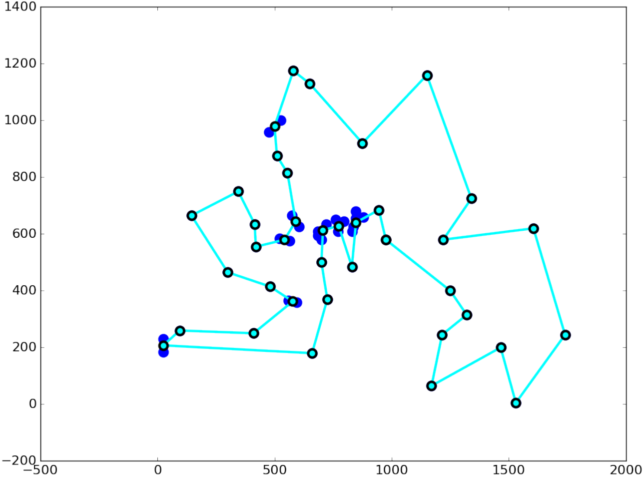
animation of the pair-center tour algorithm



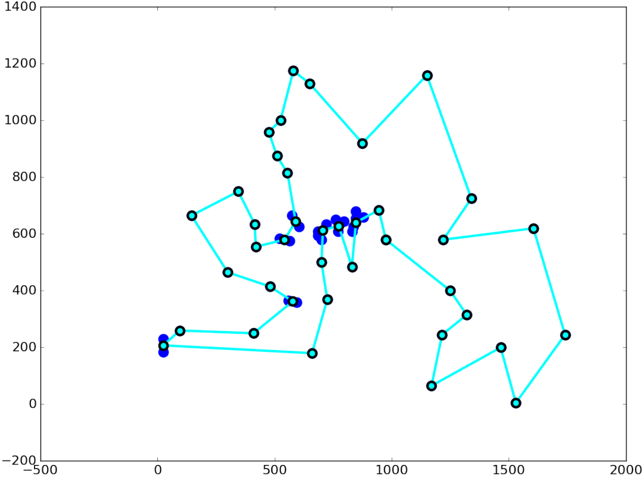
animation of the pair-center tour algorithm



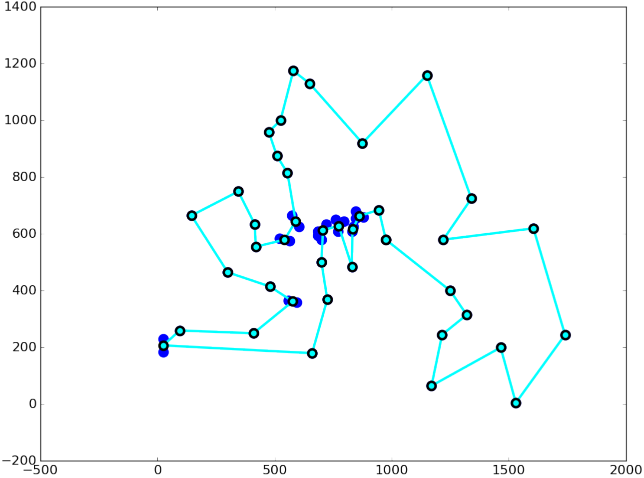
animation of the pair-center tour algorithm



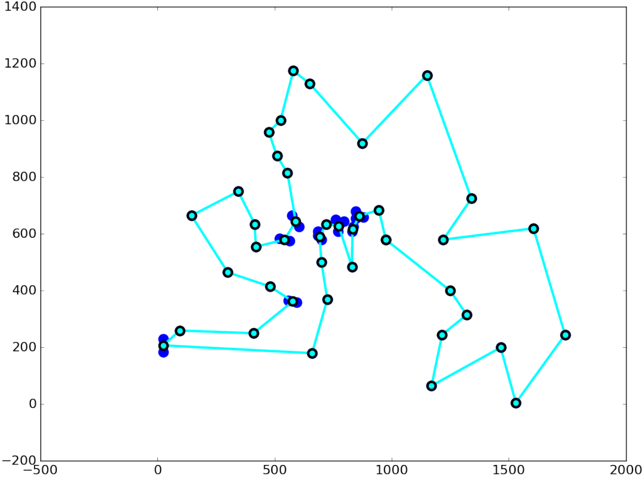
animation of the pair-center tour algorithm



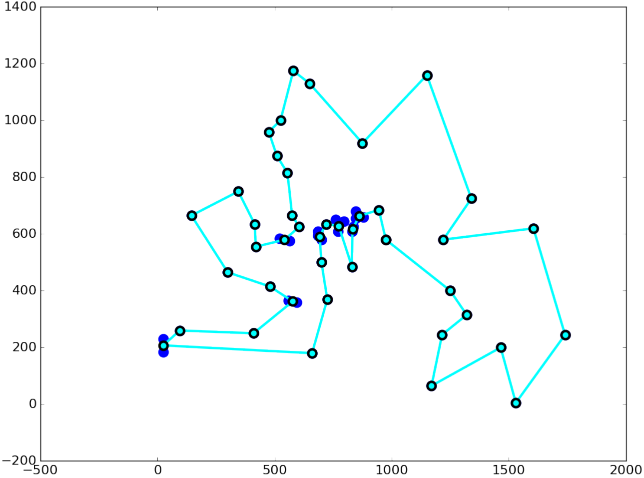
animation of the pair-center tour algorithm



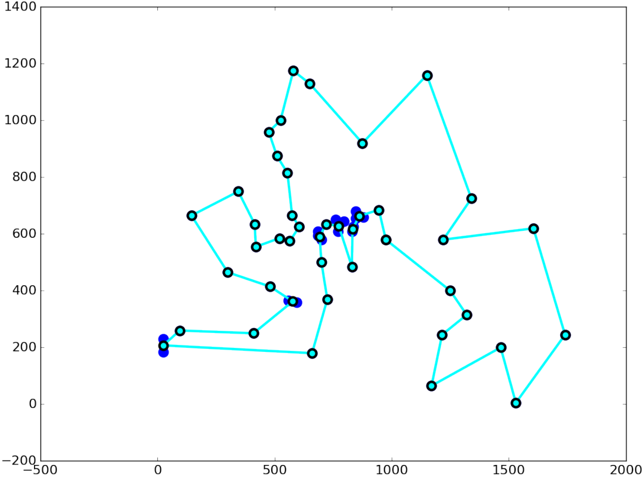
animation of the pair-center tour algorithm



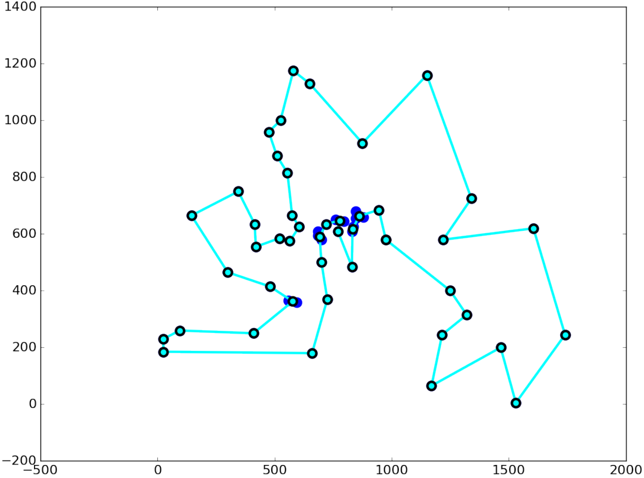
animation of the pair-center tour algorithm



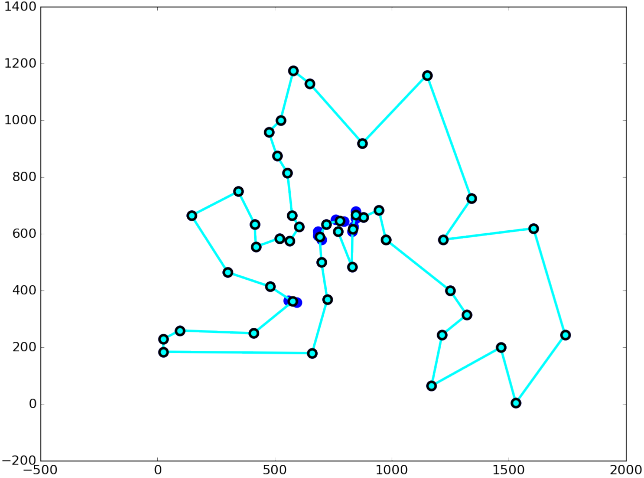
animation of the pair-center tour algorithm

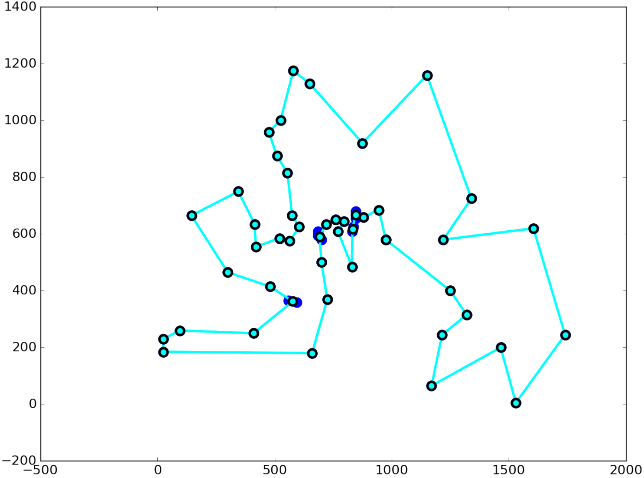


animation of the pair-center tour algorithm

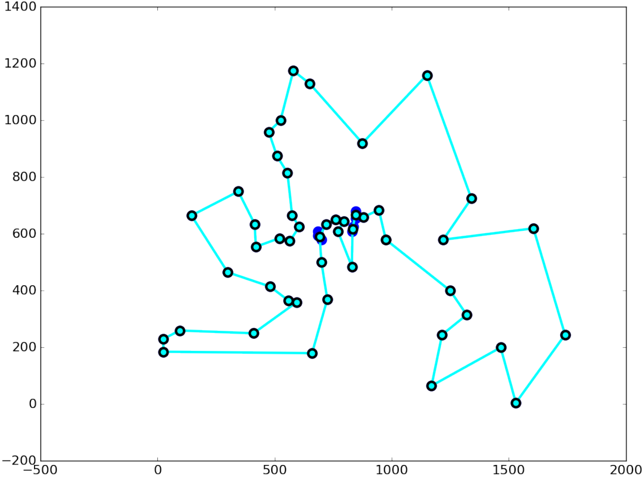


animation of the pair-center tour algorithm

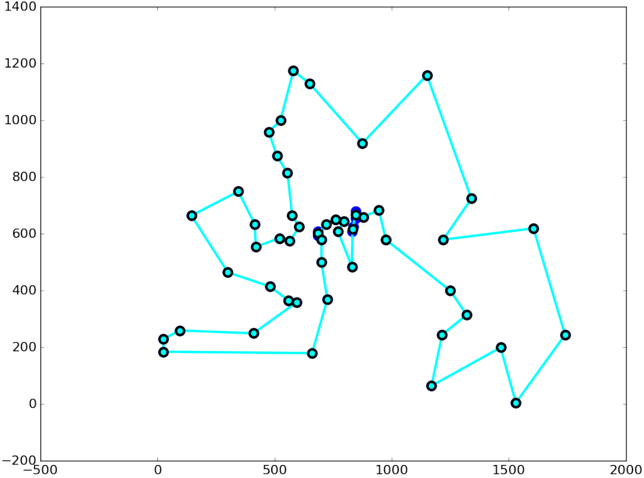



1400
1200
1000
800
600
400
200
0
−200
−500
0
500
1000
1500
2000

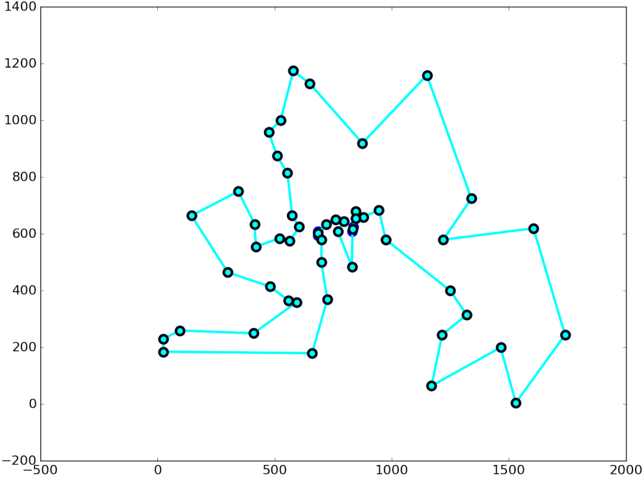
animation of the pair-center tour algorithm



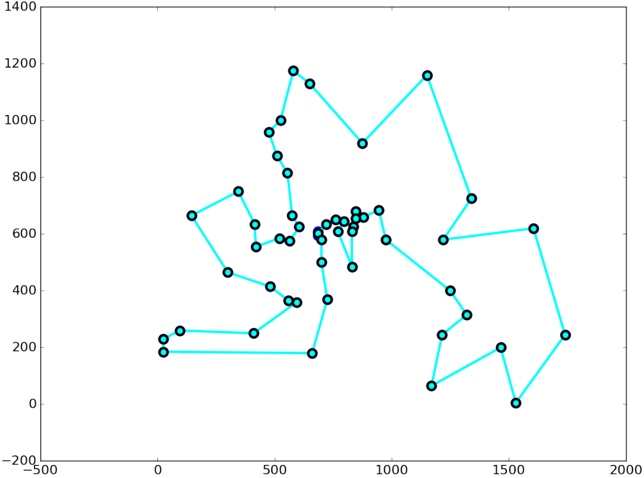
animation of the pair-center tour algorithm



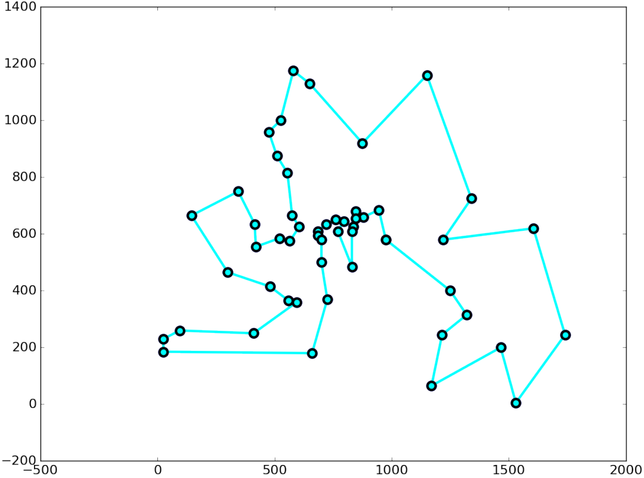
animation of the pair-center tour algorithm



animation of the pair-center tour algorithm



animation of the pair-center tour algorithm



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 (1971)*
L_{MDB}	*minimal distance bound (possible $L_{MDB} = L_{opt}$)*