

# Theoretical basis of evolutionary algorithms

Andrzej Jaskiewicz

# Biologically inspired algorithms and more...

- **Evolution-based**

- Differential evolution, Evolution strategies, Genetic/evolutionary algorithms, Genetic programming...

- **Swarm-based**

- Ant colony optimization, Artificial Bee Colony, Bat Algorithm, Crow search algorithm, Cuckoo search, Firefly Algorithm, Flower Pollination Algorithm, Grey Wolf Optimizer, Krill Herd Algorithm, Moth-Flame Optimization Algorithm, Particle Swarm Optimization, Social Spider Optimization, Whale Optimization Algorithm...

- **Physics-based**

- Electromagnetism-like mechanism, Gravitational Search Algorithm, Simulated annealing, Sine Cosine Algorithm, States of matter search...

- **Human-based**

- Fireworks Algorithm, Harmony Search, Imperialist Competitive Algorithm, Tabu search...

# The ZOO of (nature-inspired) metaheuristics

- Fausto, Fernando; Reyna-Orta, Adolfo; Cuevas, Erik; et al., From ants to whales: metaheuristics for all tastes, ARTIFICIAL INTELLIGENCE REVIEW Volume: 53 Issue: 1 Pages: 753-810 Published: JAN 2020
- Fathollahi-Fard, Amir Mohammad; Hajiaghaei-Keshteli, Mostafa; Tavakkoli-Moghaddam, Reza, Red deer algorithm (RDA): a new nature-inspired meta-heuristic, SOFT COMPUTING Volume: 24 Issue: 19 Pages: 14637-14665 Published: OCT 2020
- Torabi, Shadi; Safi-Esfahani, Faramarz, Improved Raven Roosting Optimization algorithm (IRRO), SWARM AND EVOLUTIONARY COMPUTATION Volume: 40 Pages: 144-154 Published: JUN 2018

# The problem of language (lingo)

- Solution – individual, genotype, ant, worm, raven...
- Metaphors (biological, social, physical) make it easier to understand/remember individual algorithms, but hide their similarities
- Is it possible to present a common scheme of all/most EAs/MHs?

# Iterative improvement

Generate and evaluate the initial population **X**

**repeat**

Based on population **X** and its evaluation, generate and evaluate population **X'**

From  $\mathbf{X} \cup \mathbf{X}'$  select a new population **X**

**until** stopping conditions are met

# Iterative improvement

- The population may be:
  - Singleton – local search, iterative local search, simulated annealing, Tabu search,...
  - Multi-element – evolutionary algorithm, ant colony algorithms, population algorithms in general
- What about the Tabu list? Long-term memory? Pheromone trails?

# Iterative improvement with memory

Initialize memory **M**

Generate and evaluate the initial population **X**

Update **M** based on **X**

**repeat**

    Based on population **X**, its evaluation, and **M** generate and evaluate population **X'**

    Update **M** based on **X'**

    From  $\mathbf{X} \cup \mathbf{X}'$  select a new population **X**

**until** stopping conditions are met

# Concepts of intensification and diversification

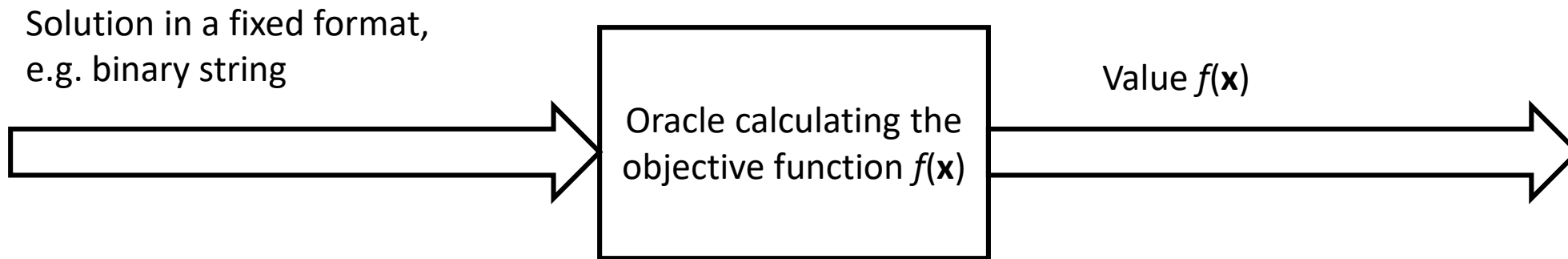
- Intensification
  - Searching for new good solutions in previously identified promising areas of the solution space
  - Usually associated with a greater focus on accepting solutions that bring improvement
- Diversification
  - Searching for new promising areas far from previously searched
  - Usually associated with the acceptance of solutions that bring deterioration
- The goal is to find a good balance between intensification and diversification
- This balance can change over time, e.g. in simulated annealing we gradually move from diversification to intensification



# Adaptation of EAs/MHs to a specific problem

- Generate an initial population  $X$
- Generate a new population  $X'$ 
  - Means using problem-specific operators (recombination, neighborhood, perturbation...) to generate new solutions
- EAs/MHs are algorithm schemes

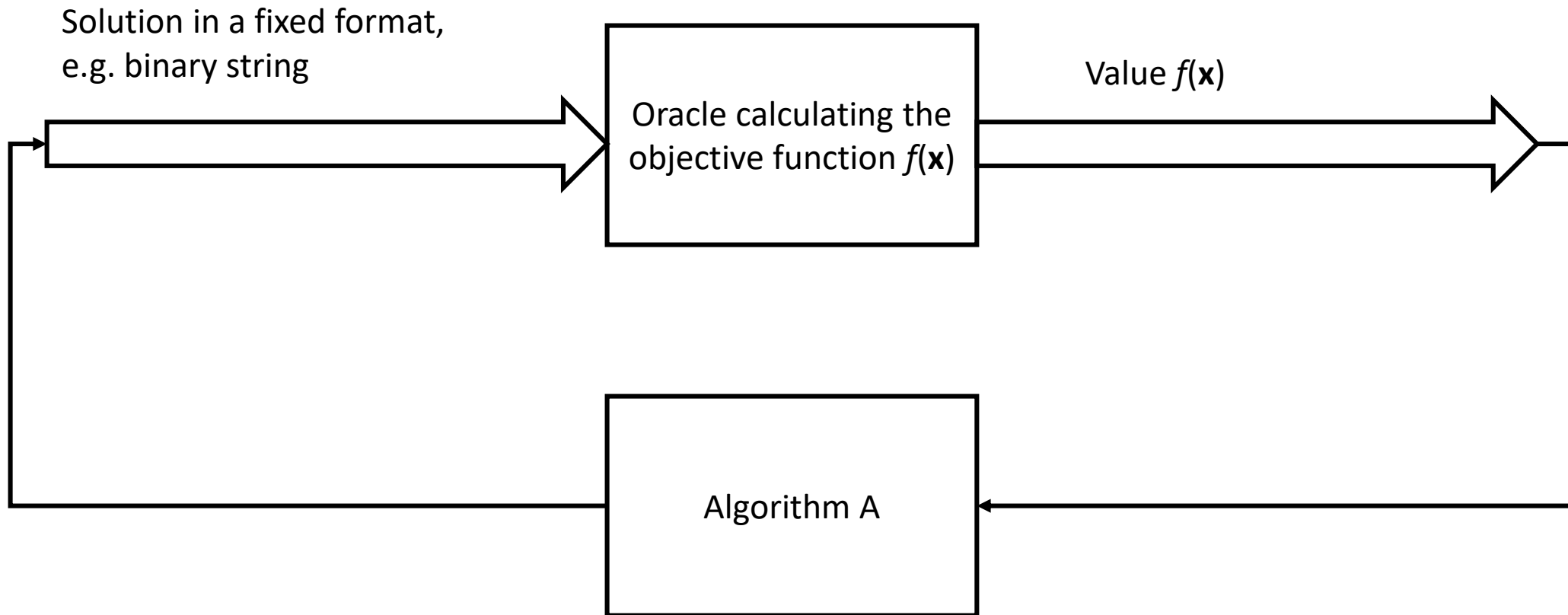
# Oracle model, black box



# Oracle – a function with an unknown implementation

```
double function (Vector<boolean>) {  
    // Unknown implementation  
    ...  
    return...  
}
```

# Optimization of the function contained in the oracle, black-box optimization



# Questions

- What can we expect by using some algorithm A to optimize the function contained in the oracle?
- For example, will an iterative local search work better than a random search?
- In local search, is it better to accept solutions that bring improvement than deterioration? It seems obvious.

# No free lunch theorem

- Assumptions:

1. The solution space is finite (e.g. a binary vector of constant length)
2. The number of possible values of the function is finite
3. The oracle acts deterministically returning the result of a certain function  
From 1, 2, and 3 it follows that the number of functions that can be in the oracle is finite
4. Each of the possible functions has an equal probability of occurring in the oracle (!)
5. We do not have any additional knowledge about the functions  $f(\mathbf{x})$
6. Algorithm A generates solutions without repetitions

# No free lunch theorem

- **Under the above assumptions, each algorithm A will give in a given number of iterations the same expected value of any "meaningful" measure of the quality of the solutions obtained..**
- A "meaningful" measure is a measure based on the distribution of values of all solutions generated by algorithm A.
- Eg.:
  - the best generated solution
  - average value of N best solutions
  - average value of all generated solutions
- but not:
  - average value of solutions in the current population
  - average value of solutions in the final population
  - solution returned by the algorithm

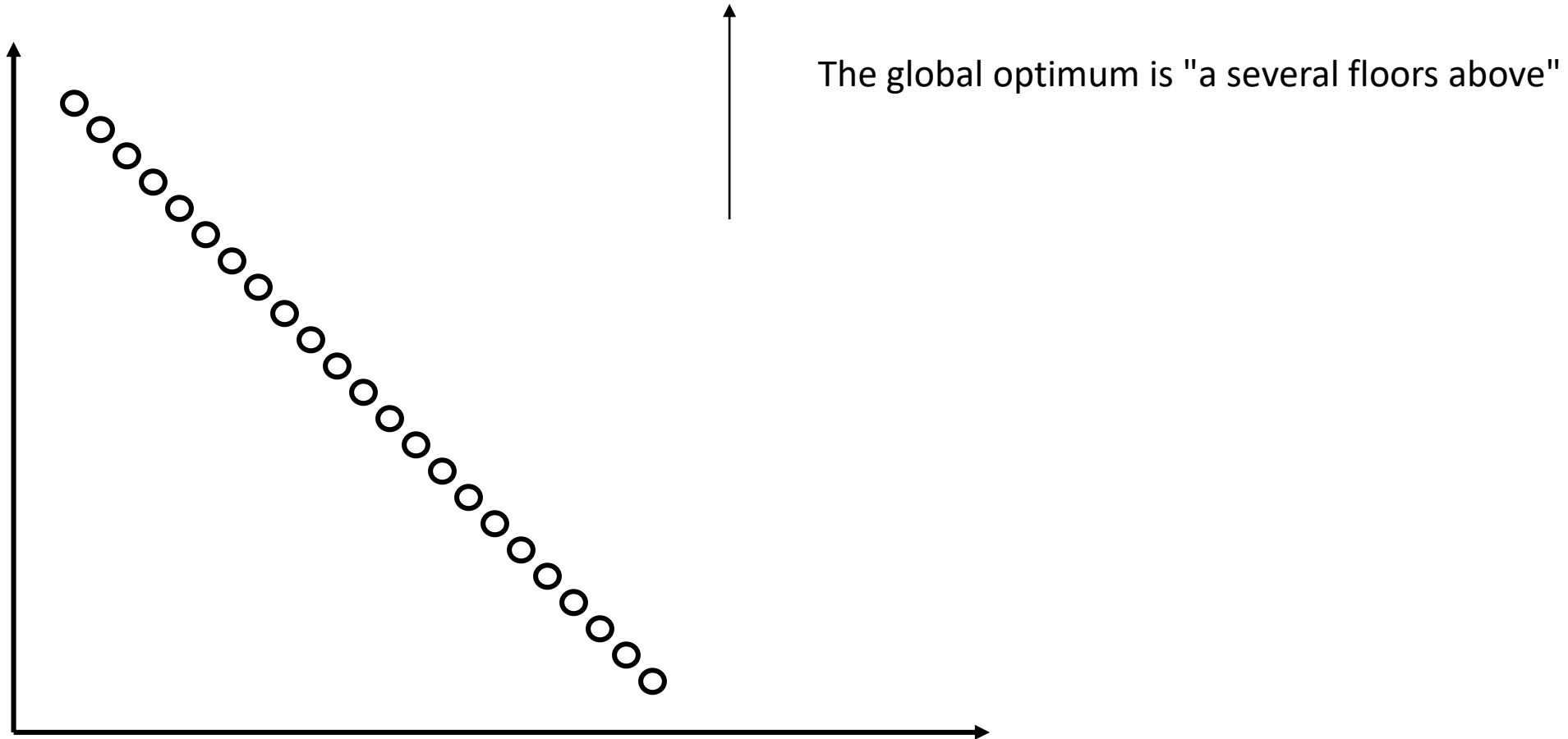
(because the values of different generated solutions can be taken into account differently number of times)

# No free lunch theorem

- „Each" means **each**
  - For example, a local search that selects the best solution from the neighborhood will give the same expected value to any meaningful measure of quality as a local search that selects the worst solution from the neighborhood
- If algorithm A works better than algorithm B on certain functions, then there must be other functions on which it will perform worse



# Example of a "deceptive" function for local search - maximization



# Algorithm A as a rule for generating new solutions

- Solution generated in the  $i$ th iteration:

$$\mathbf{x}_i = A(\mathbf{x}_1, f(\mathbf{x}_1), \mathbf{x}_2, f(\mathbf{x}_2), \dots, \mathbf{x}_{i-1}, f(\mathbf{x}_{i-1}))$$

- In other words, the maximum possible memory **M** is a list of all previously generated solutions and their values of the objective function

# Outline of the proof - intuition

- Consider two binary variables  $x_1, x_2$ .
- Four possible solutions:
  - 00
  - 01
  - 10
  - 11
- The domain of the objective function is  $\{0, 1\}$  – only two values

# All possible objective functions

$x_1$	$x_2$		f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16
0	0		0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1		0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0		0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1		0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

For example:

$$f1 = 0$$

$$f2 = x_1 \wedge x_2$$

$$f4 = x_1$$

$$f5 = x_2$$

$$f8 = x_1 \vee x_2$$

$$f13 = \neg x_1$$

# The first step of the algorithm

$x_1$	$x_2$		f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16
0	0		0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1		0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0		0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1		0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

For each of the possible four solutions, we have the distribution of the values of the objective function

We generate a solution  $\mathbf{x} = [0,0]$

We receive from the oracle  $f(\mathbf{x}) = 0$

What does this mean?

# Knowledge gained after the first step

$x_1$	$x_2$		f1	f2	f3	f4	f5	f6	f7	f8
0	0		0	0	0	0	0	0	0	0
0	1		0	0	0	0	1	1	1	1
1	0		0	0	1	1	0	0	1	1
1	1		0	1	0	1	0	1	0	1

We generate a solution  $\mathbf{x} = [1, 0]$

We receive from the oracle  $f(\mathbf{x}) = 0$

What does this mean?

# Second step

$x_1$	$x_2$		f1	f2	f3	f4	f5	f6	f7	f8
0	0		0	0	0	0	0	0	0	0
0	1		0	0	0	0	1	1	1	1
1	0		0	0	1	1	0	0	1	1
1	1		0	1	0	1	0	1	0	1

For each of the possible four solutions, we have the distribution of the values of the objective function

We generate a solution  $\mathbf{x} = [1, 0]$

We receive from the oracle  $f(\mathbf{x}) = 0$

What does this mean?

# Knowledge gained after the second step

$x_1$	$x_2$		f1	f2		f5	f6
0	0		0	0		0	0
0	1		0	0		1	1
1	0		0	0		0	0
1	1		0	1		0	1



# Proof of the No free lunch theorem

- The proof consists in showing that in each step of any algorithm, regardless of the choice of previous solutions, we have the same distribution of objective function values for all possible solutions to be generated

# The assumption of the theorem can be weakened

- It is sufficient to assume that a subset of possible functions is closed under permutation (c.u.p.), i.e. for any permutation of variables the set of functions remains unchanged

# The analogous No free lunch theorem also exists for machine learning

- Relationships between ML and optimization (ML is basically the optimization of some measure of error)
- Analogies
  - solutions - objects, cases, examples
  - generated solutions – learning examples
  - value of the objective function - decision class
  - rule/algorithm - classification mechanism
  - knowledge - knowledge
- Differences
  - Objective: ML - classification, optimization – generation of new solutions
  - ML: assumption discriminatory description, optimization probably irrelevant
  - Optimization – ordered values of the objective function

# The NFL Theorem and "Practice"

- How does the NFL theorem relate to the knowledge that typical EA/metaheurists work well in practice?
- "In theory there is no difference between theory and practice, but in practice there is"

# What assumptions of NFL may be not met in practice?

- We do not have any additional knowledge about the functions  $f(x)$ 
  - Knowledge about the function can be used in operators (neighborhood, perturbation, recombination) – e.g. the operator can be guided by the value of the (components of) objective function – as a result, a dedicated method
  - For example, in TSP, using a list of nearest vertices to define candidate moves in a local search...
  - ... but LS for TSP works well even without candidate movements

# What assumptions of NFL may be not met in practice?

- The set of functions is c.u.p. and each of the possible functions has an equal probability of occurring in the oracle
  - For example, a specific combinatorial problem may not allow all functions (many functions do not correspond to any instances of this problem) and the set of allowed functions does not have to be c.u.p.
    - The share of c.u.p subsets of functions among all possible subsets quickly tends to zero as the size of the solution space increases. ...
    - ...that is, intuitively, if we are dealing with a subset of all functions, it is almost certainly not c.u.p. and NFL does not apply
  - The distribution of the values of the problem parameters can translate into not uniform probability distribution of individual instances and functions

# An example of a simple combinatorial problem

$$f(\mathbf{x}_1, \mathbf{x}_2) = w_1 \mathbf{x}_1 \vee (w_2 (\neg \mathbf{x}_2))$$

$$w_1, w_2 \in \{0,1\}$$

$$w_1 = 0 \\ w_2 = 0$$

$$w_1 = 1 \\ w_2 = 0$$

$$w_1 = 0 \\ w_2 = 1$$

$$w_1 = 1 \\ w_2 = 1$$

$x_1$	$x_2$	f1	f4	f11	f12
0	0	0	0	1	1
0	1	0	0	0	0
1	0	0	1	1	1
1	1	0	1	0	1

Individual solutions differ in the distribution of values of the objective function

The probability distribution of parameters can affect the probability distribution of functions

- $P(w_1=1) = 0.7, P(w_2=1) = 0.7$

- $P(f = f_1) = 0.09$

- $P(f = f_4) = 0.21$

- $P(f = f_{11}) = 0.21$

- $P(f = f_{12}) = 0.49$



In practice, the speed of generating solutions may also be important

- The NFL theorem refers to the number of solutions generated, not to running time
- If a given method generates new solutions faster (because, for example, they differ less than the previous ones), then in practice it will be better

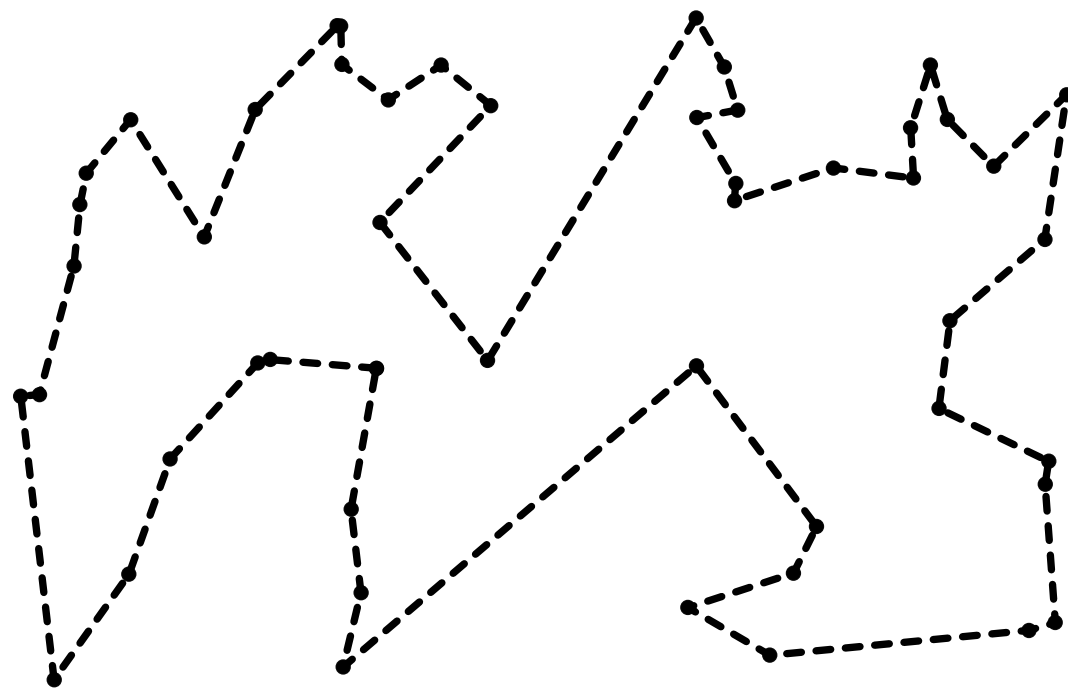
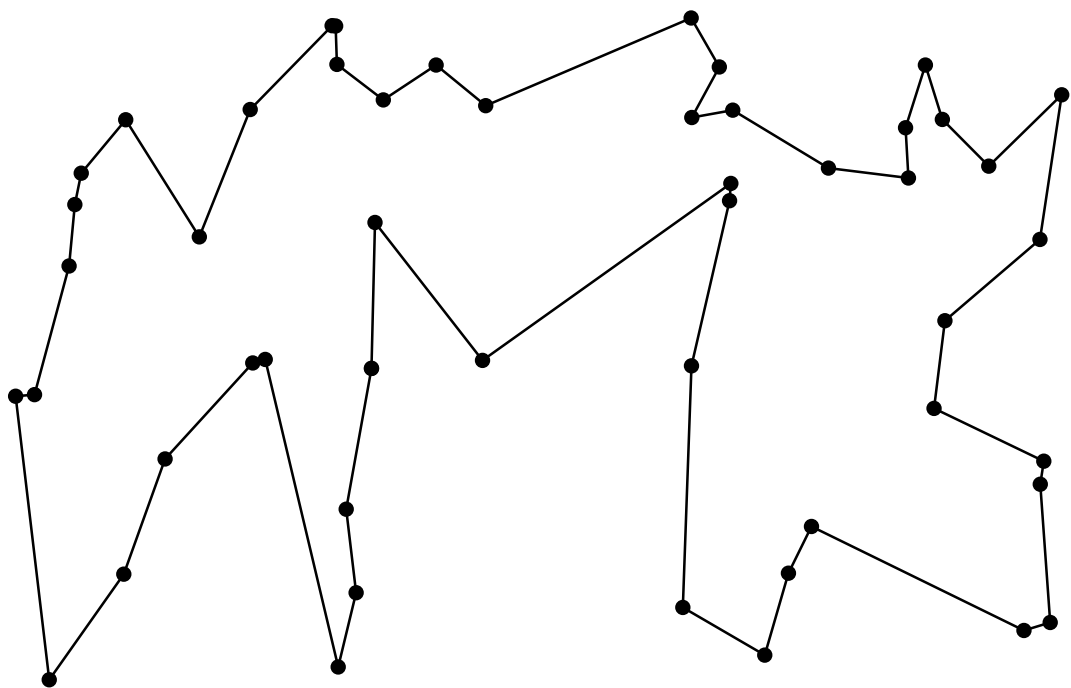
# What are EAs/MHs

- These are not universal methods of black-box optimization, because according to the NFL such methods do not exist
- They work well on a certain class of functions (problems) that apparently often occur in "practice,,
- What features do these functions (problems) have?

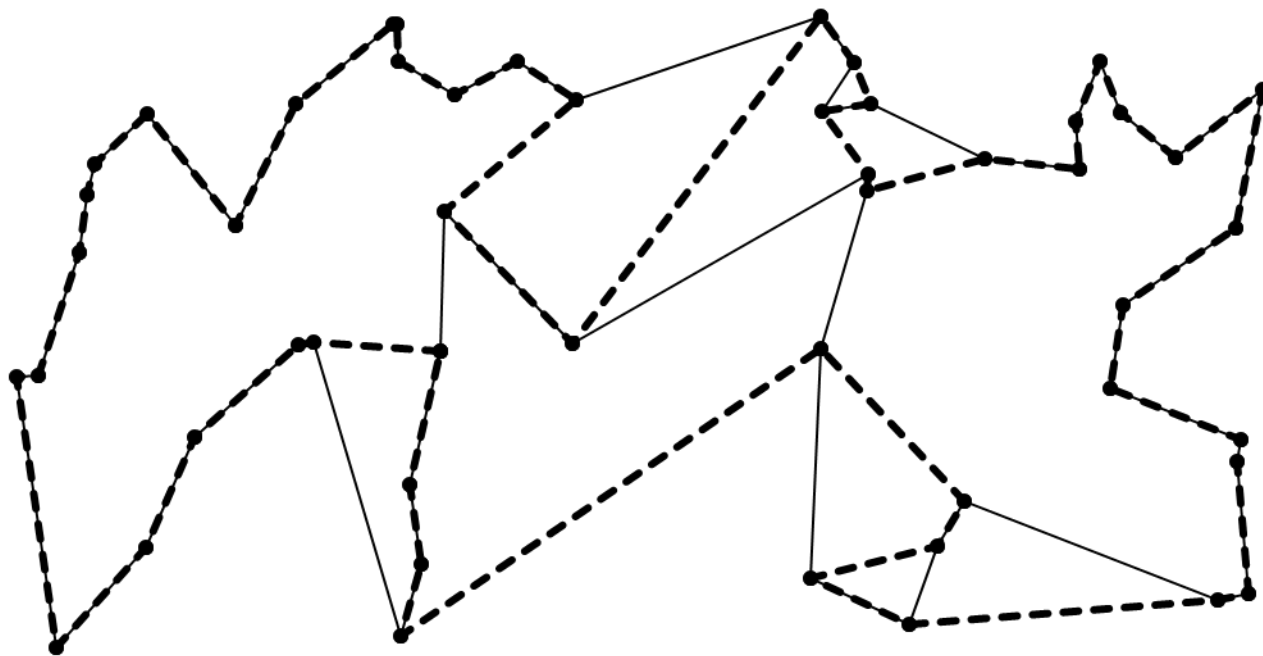
# Boese, Kahng and Muddu experiment (1994) for TSP

- Typical EAs/MHs work very well on the traveling salesman's problem
- Instances of 100 and 500 cities
- 2500 "random" local optima obtained by local search in a greedy version starting from random initial solutions
- Measure of similarity of solutions – number of common edges
- Study of the relationship between the mutual similarity of local optimas (average to all others or to the best) and their quality
- Observations
  - Very strong correlations between the similarity of solutions and their quality – better are more similar
  - Stronger correlations for average similarity to all others than to the best

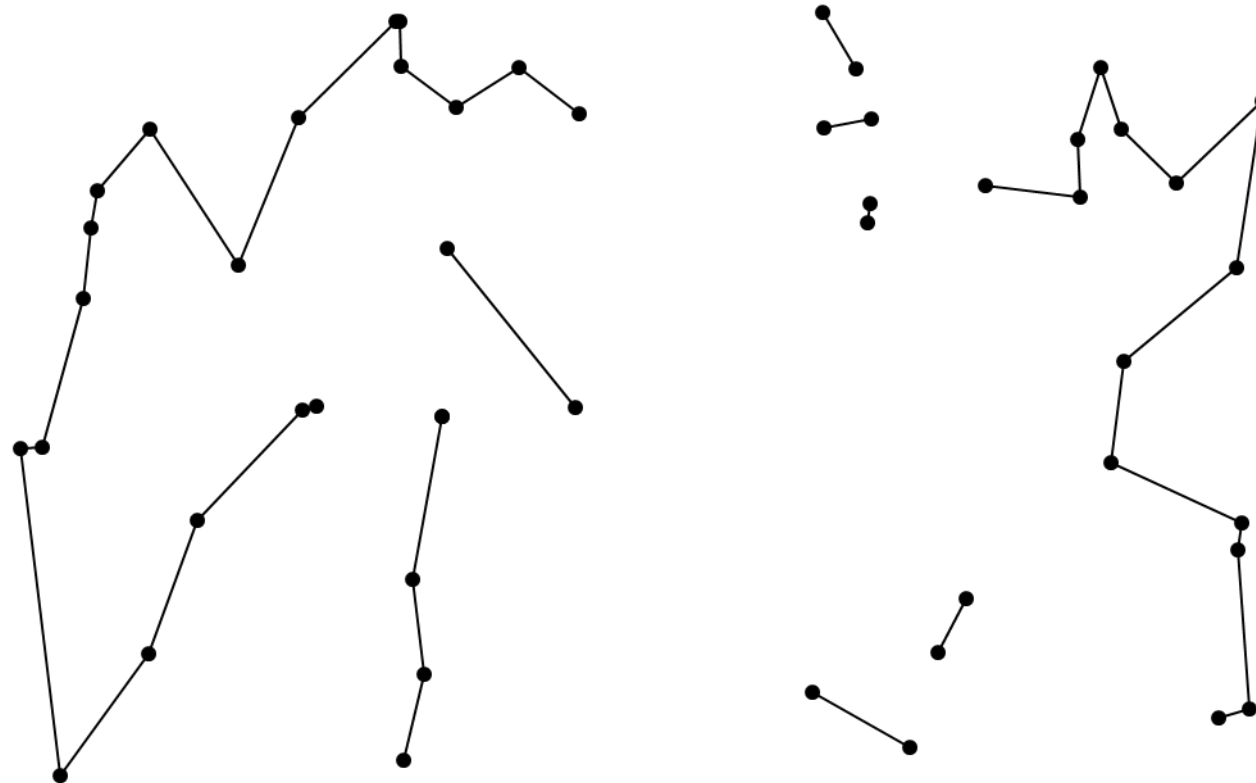
# Two exemplary local optima of TSP



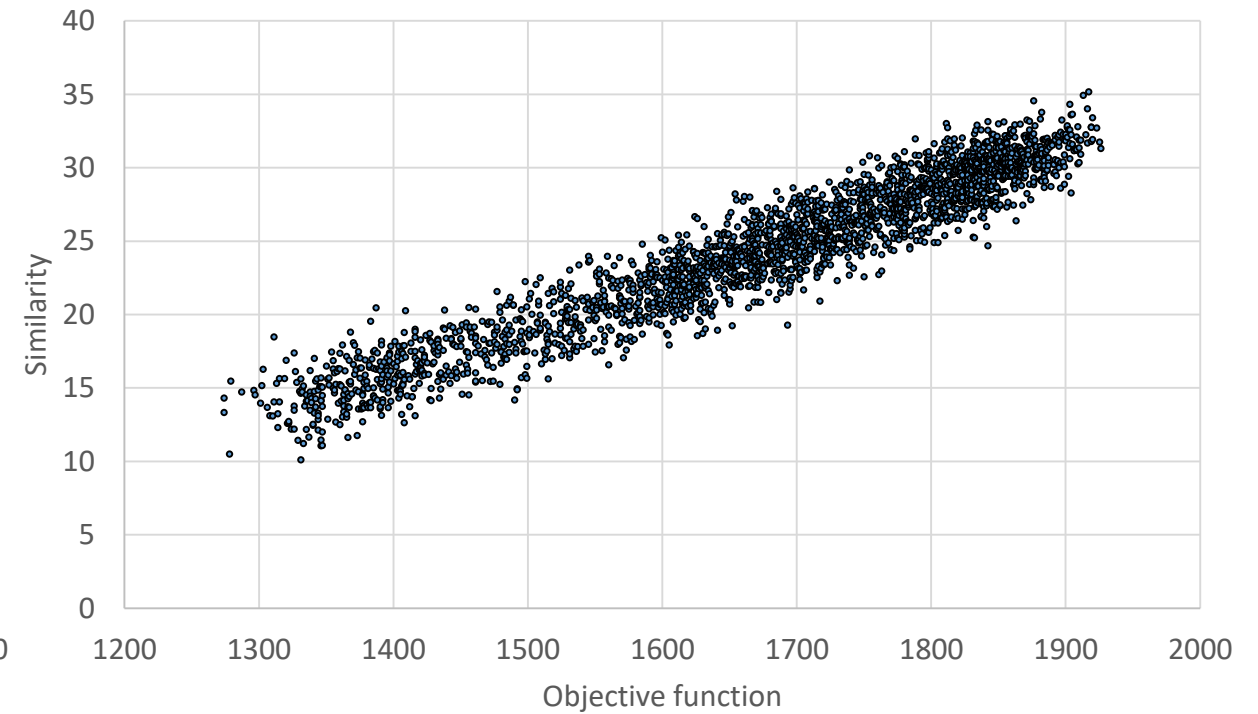
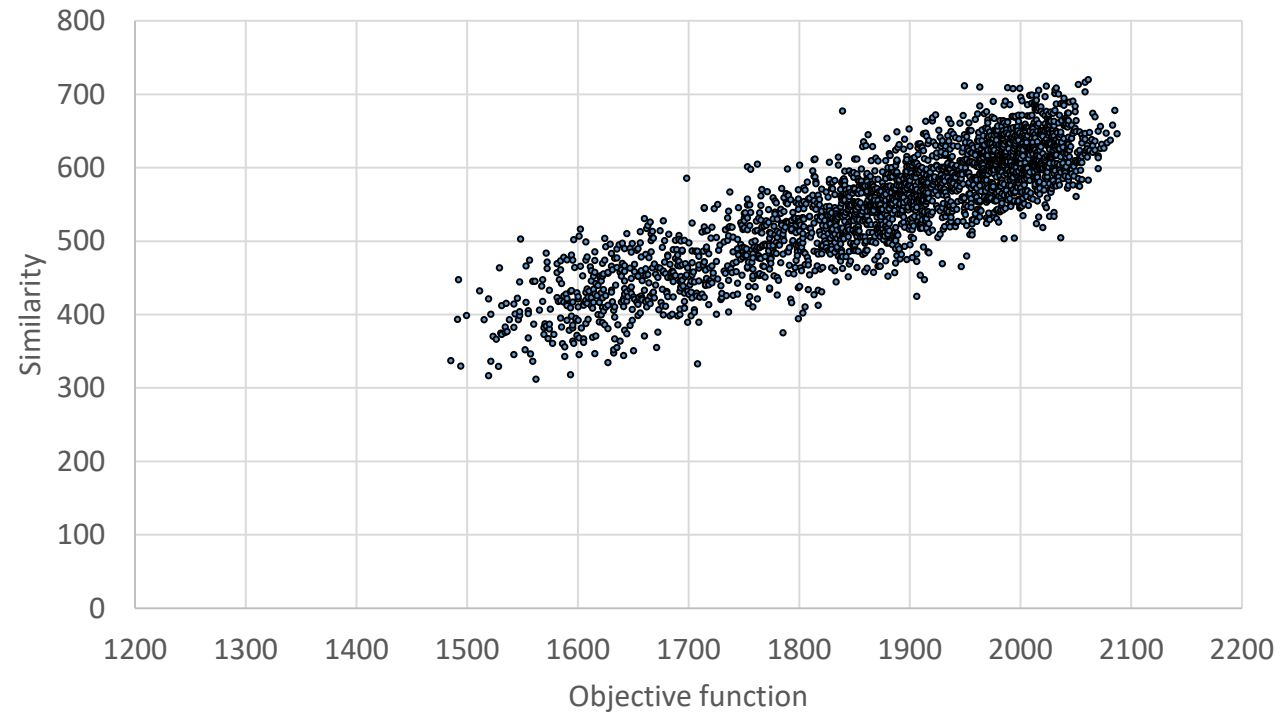
# Two exemplary local optima of TSP



# Common edges (and paths)



# Correlation examples (other problem and experiment) – maximized goal function



# Other observations of Boese, Kahng and Muddu (1994) for TSP

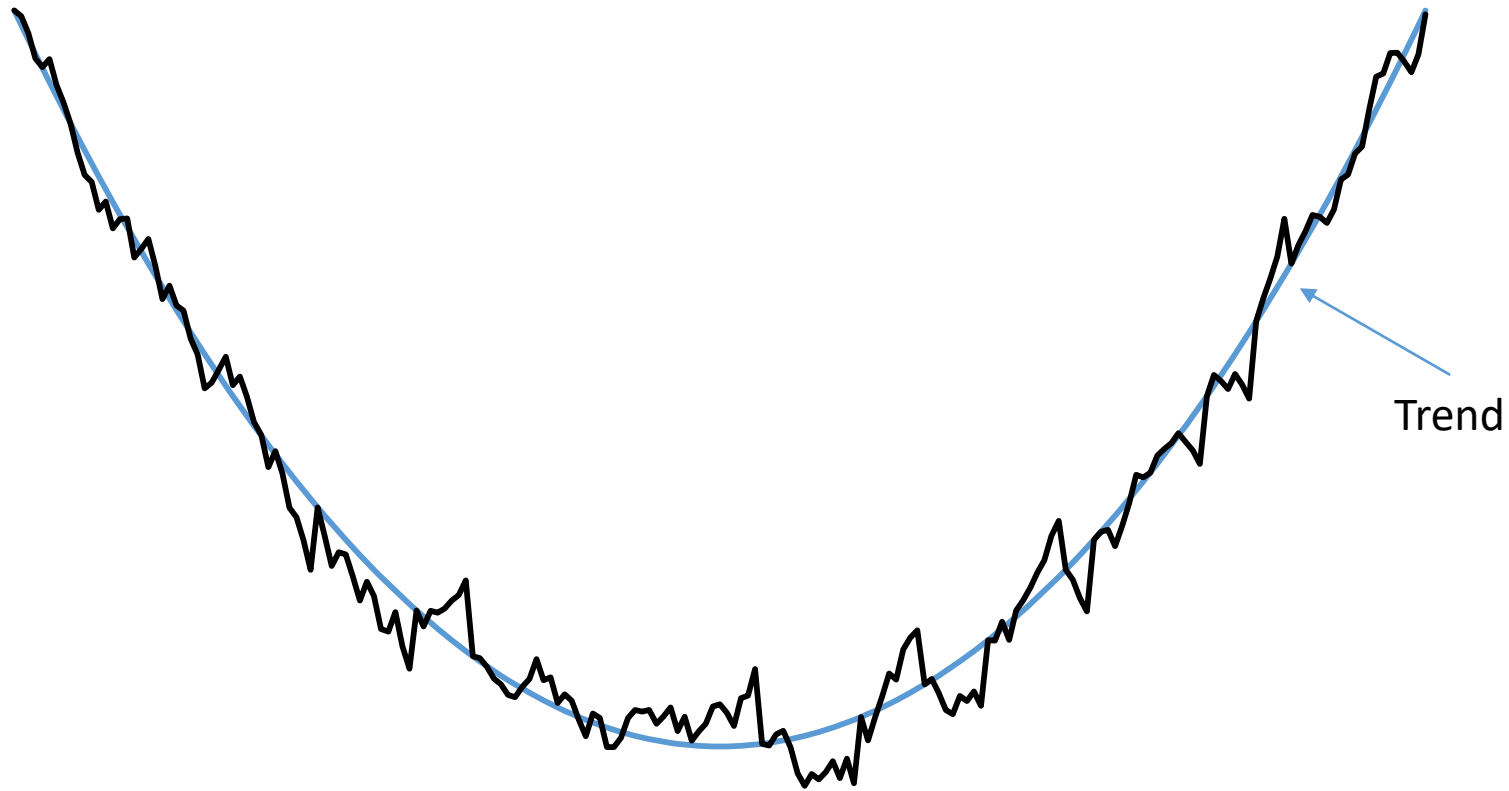
- The average expected distance of two random solutions is  $N-2$  (two common edges)
- For 100 cities, the maximum distance between local optima was 48 (52 common edges)
- All local optima are contained in a "sphere" with a diameter of 48. The sphere includes  $1/10^{59}$  solutions
- For 500 cities: diameter of the "sphere" 243,  $1/10^{468}$  solutions



# Global convexity of function (problem)

- The authors named the observed feature a global convexity or deep valley
- In mathematics, a function is globally convex if it can be reduced to a convex function by adding/subtracting a certain value  $\leq \varepsilon$  to its value for individual solutions
- It can be understood as a combination of trend (convex function) and (deterministic) noise
- Solutions that lie close to the optimum of the trend are on average better and more similar to other good solutions (which lie around the optimum of the trend)

# Example of a globally convex function of one variable



# Comments

- Global convexity means that good solutions are similar to each other, not that solutions similar to good ones are also good (noise can be very large and small differences can cause large jumps in the value of the objective function)
- Global convexity depends on how the similarity of solutions is measured

# What is the source of the trend for TSP?

- Good solutions consist of short edges (trend)...
- ... but  $N$  of the best edges is unlikely to form a Hamiltonian cycle
- ... to create a Hamiltonian cycle, we need to replace certain edges with longer ones, i.e. increase the value of the objective function – add a deterministic noise

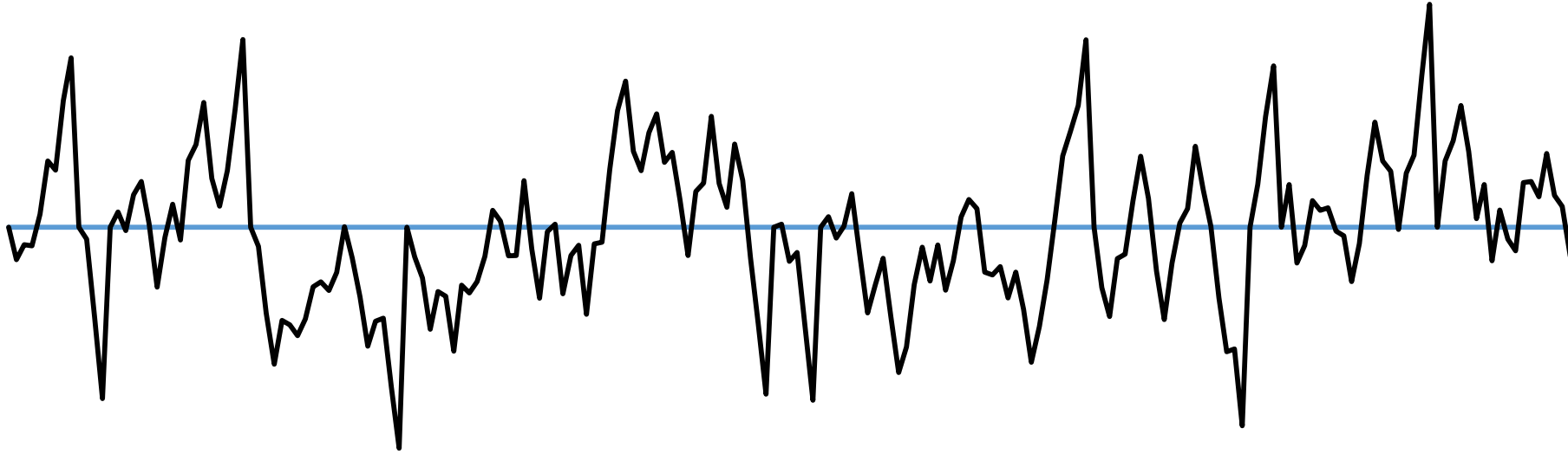
# Global convexity in other problems

- Global convexity has been demonstrated for many other problems, e.g.:
  - Graph partitioning - Boese, Kahng and Muddu (1994), quadratic assignment - Merz, Freisleben (2000), vehicle routing - Jaskiewicz, Kominek (2000, 2003), vehicle routing - Kubiak (2004), Capacitated Vehicle Routing Problem (CVRPTW), Pickup and Delivery Problem (PDPTW), and Team Orienteering Problem (TOPTW) with time windows - Cybula, Jaskiewicz, Pełka, Rogalski, Sielski (2021)
  - Intuitive trends for many problems
  - For example, in the VRP problem, we expect the occurrence of short but also "well directed" (to/from the base) edges. Edges should also connect vertices with matched time windows

# How global convexity explains EAs/MHs

- Iterative local search, large-scale neighborhood search, simulated annealing, tabu search... – a model of tossing a ball in a rough bowl
- Recombination operators – construction of new solutions combining the features (and therefore similar to) parents (throwing ropes over the valley)
  - The similarity to good solutions does not guarantee high quality, but:
    - the average quality will be higher
    - Local search may be able to improve such a solution very quickly (there are other very good solutions nearby)

# Globally "flat" problems



- No global trend
- Local search may still work, although the best method may be a simple multiple start local search
- On the one hand, difficult for EAs/MHs, on the other hand, we can not lose too much on the value of the objective function

# Other studies

- There hasn't been much work on the question „for what functions do EAs/MHs work?", but a lot of work on "what functions are simple/difficult for EAs/MHs?"



# Fitness distance correlation FDC

- Approach similar to global convexity testing – quality-similarity/distance correlation study
- Distance from the global optimum
- Originally, 0-1 encoding was assumed

# Holland schema theorem

- Short, low-order and well-adapted schema spread through subsequent generations according to the exponential law of growth
- The occurrence of a common schema in the crossed solutions is related to their similarity

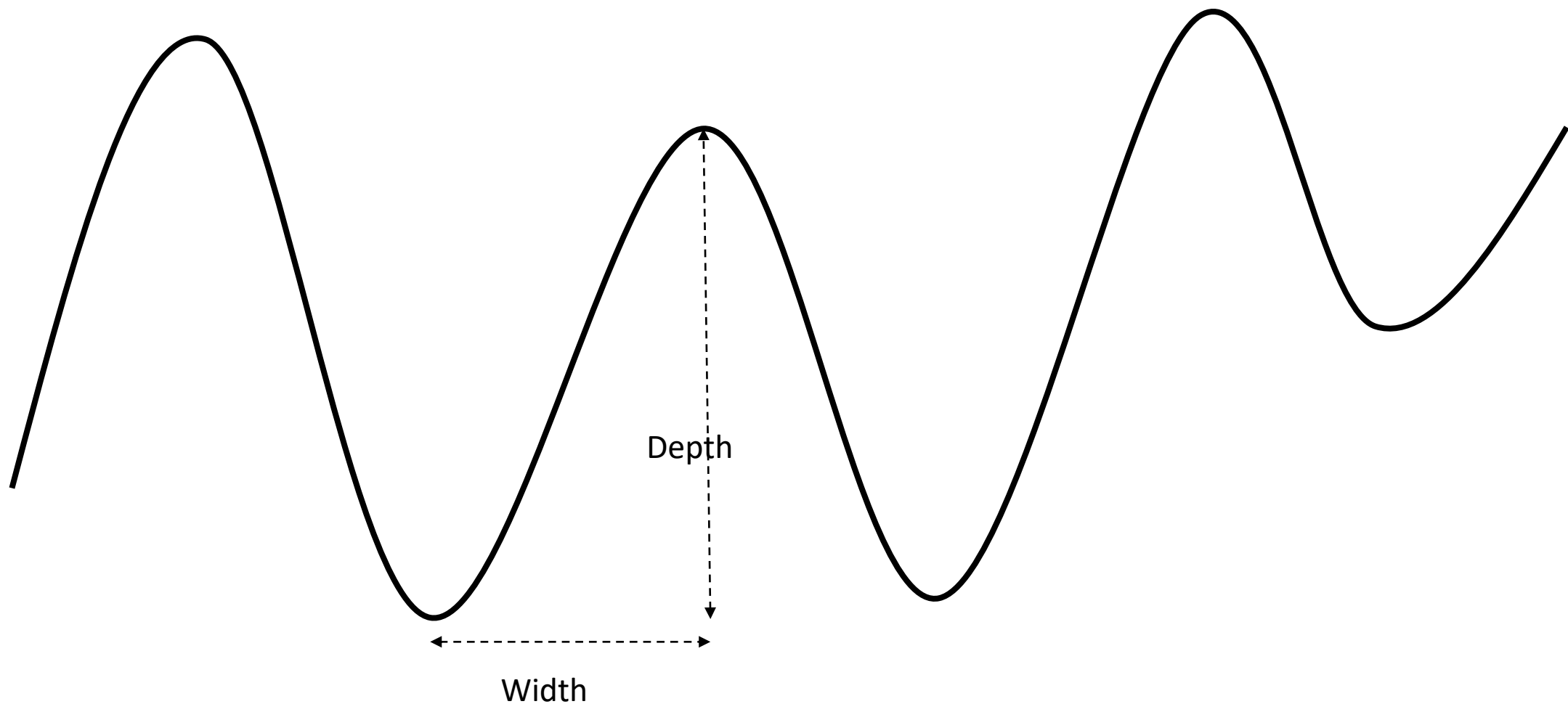
# Entropy in the set of local optimas

- Low entropy – local optima are similar to each other
  - Global convexity means low entropy
- High entropy – local optima freely scattered in the solutions space – no trend

# Local features

- Efficiency of local search relative to random search
  - For TSP, the local optimum with two edge exchange move is achieved in average  $n$  iterations, where  $n$  is the number of cities, i.e. complexity is  $O(n^3)$
  - Meanwhile, for  $n=100$ , local optima are contained in a sphere constituting  $1/10^{59}$  of the solutions space
  - Speed-up  $10^{59} / 10^6 = 10^{53}$
- Depth and width of local optima
  - How many moves do one needs to make on average or how much one needs to deteriorate the objective function to get to another local optimum

# Depth and width of local optima



# Ruggedness and autocorrelation coefficient

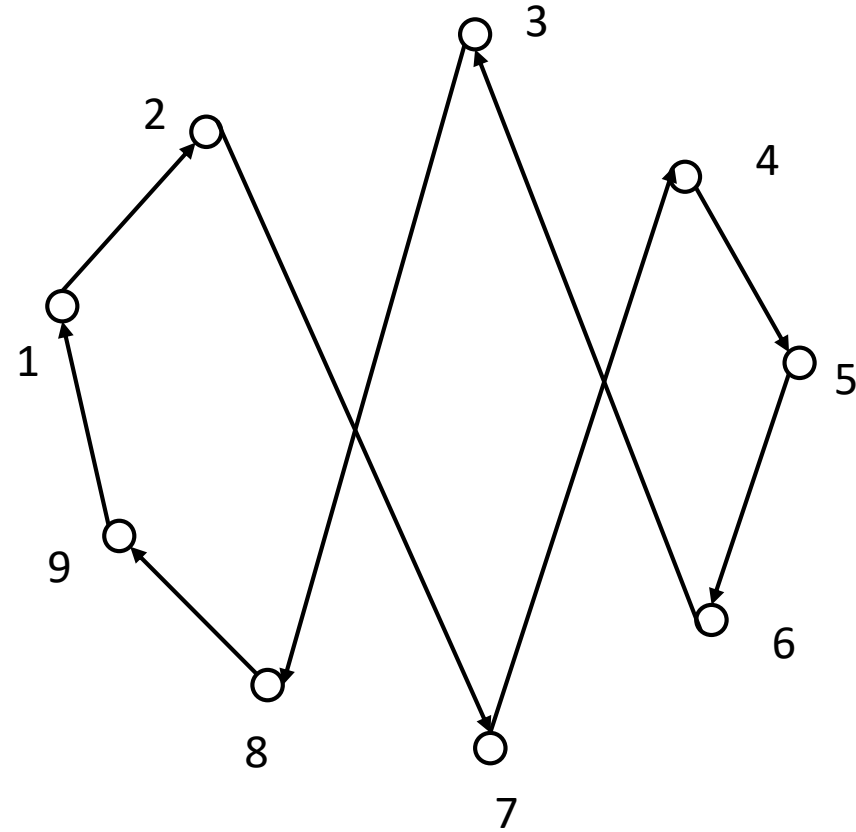
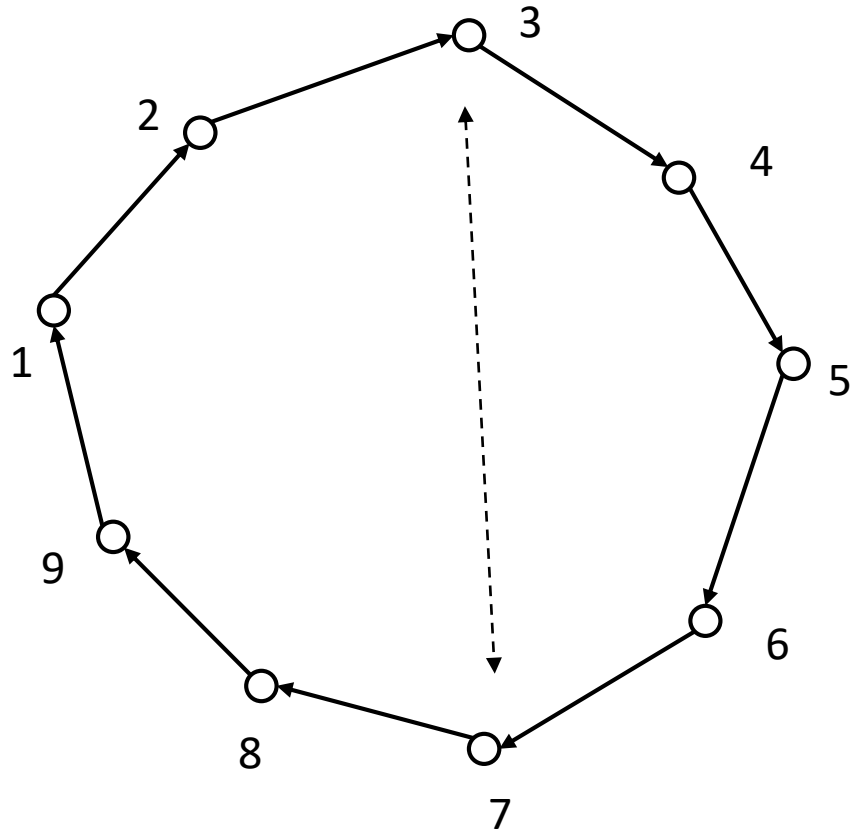
$$\rho(d) = \frac{\overline{(f(\mathbf{x}) - f(\mathbf{y}))^2 |_{d(\mathbf{x},\mathbf{y})=d}}}{\overline{(f(\mathbf{x}) - f(\mathbf{y}))^2}}$$

- $d(\mathbf{x},\mathbf{y})$  – measure of the distance of the solutions, the number of steps of local search required to go from  $\mathbf{x}$  to  $\mathbf{y}$
- $d=1$  means adjacent solutions in the numerator
- Independent from objective function scaling
- Hypothesis – local search works better for smaller autocorrelation coefficients

# Example of neighborhood operators for TSP

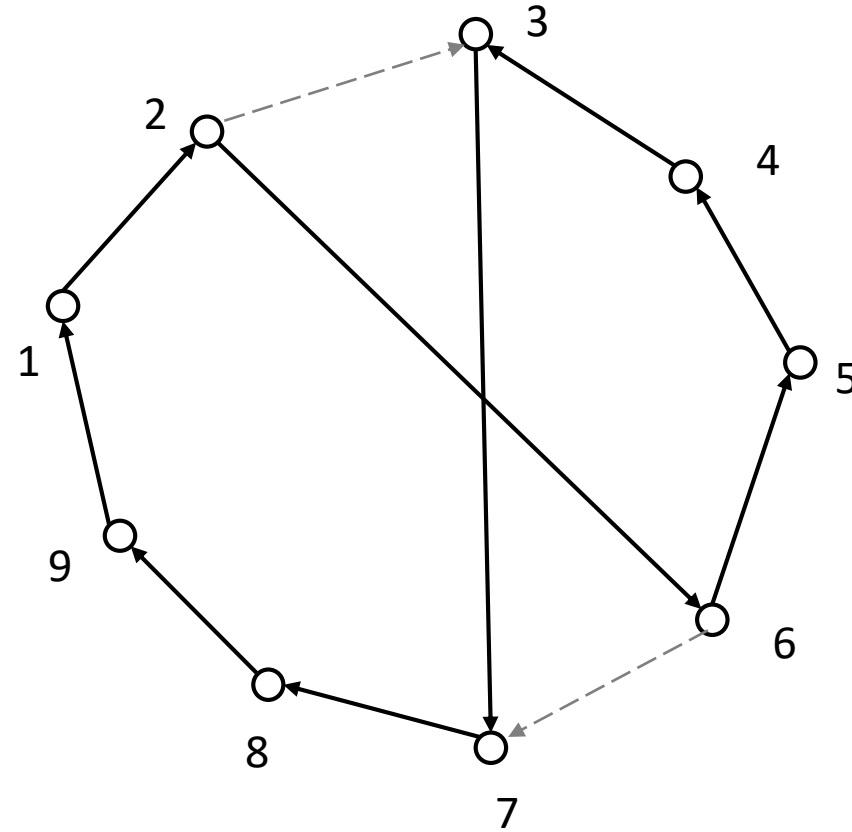
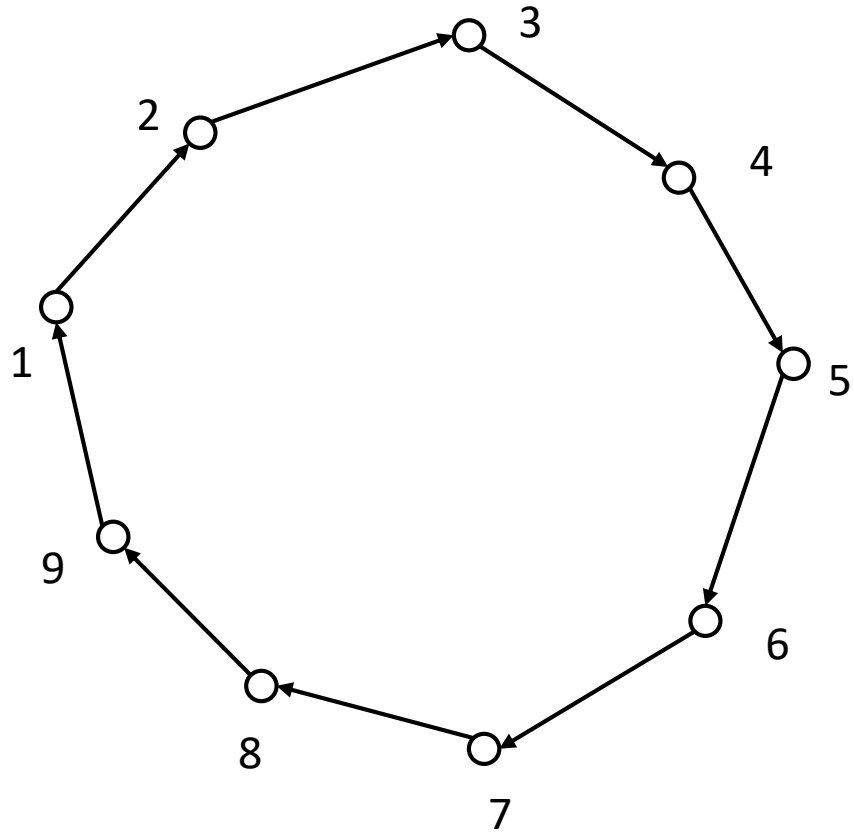
- Exchange of two vertices
  - Neighbor solutions differ in four edges
- Exchange of two edges
  - Neighbor solutions differ in two edges

# Example: Exchange of two vertices in TSP





# Exchange of two edges in TSP



# Example of neighborhood operators for TSP

- The average difference in the value of neighboring solutions is therefore greater for the exchange of two vertices
- The size of the neighborhood is similar in both cases  $O(n^2)$
- Conclusion: the exchange of two edges should work better – it is perfectly confirmed in practice
- However, the reasoning cannot be easily transferred to neighborhoods of different sizes, e.g. exchange of 3 edges will give better results (in a longer time), although the autocorrelation coefficient will be higher

# Fitness/objective function landscape analysis

- In general, this type of research is called fitness/objective function landscape analysis
- fitness/objective function landscape – triple (set of solutions, measure of distance/similarity, objective function)

# Summary – on what functions/problems do EAs/MHs work

- Global structure, trends – good solutions spread out not uniformly
- Local character (smoothness) of the objective function

# What does knowledge about the above features of problems/functions give?

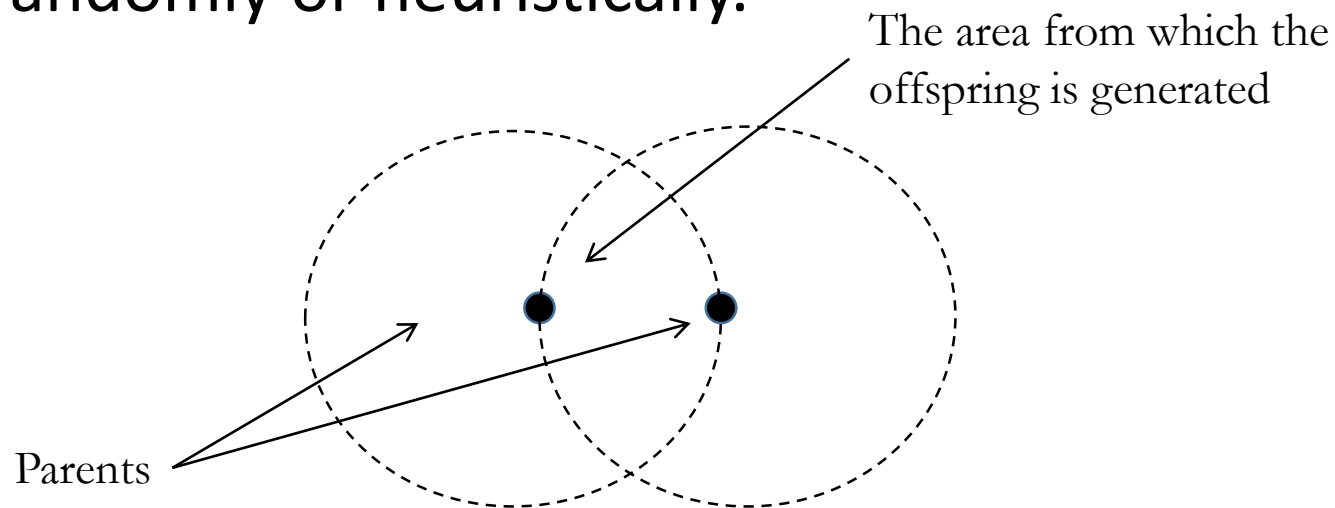
- The importance of coding, neighborhood, recombination
  - For example, global convexity/autocorrelation may depend on these choices.
  - Selection of a neighborhood with a low autocorrelation coefficient
- Methods motivated by global convexity:
  - E.g. Adaptive multiple start local search
- Distance-preserving crossover/recombination operators
- Destroy-repair operators guided by another solution(s) – removal of important different features
  - The difference between destroy-repair and recombination operators may blur

# The importance of operators

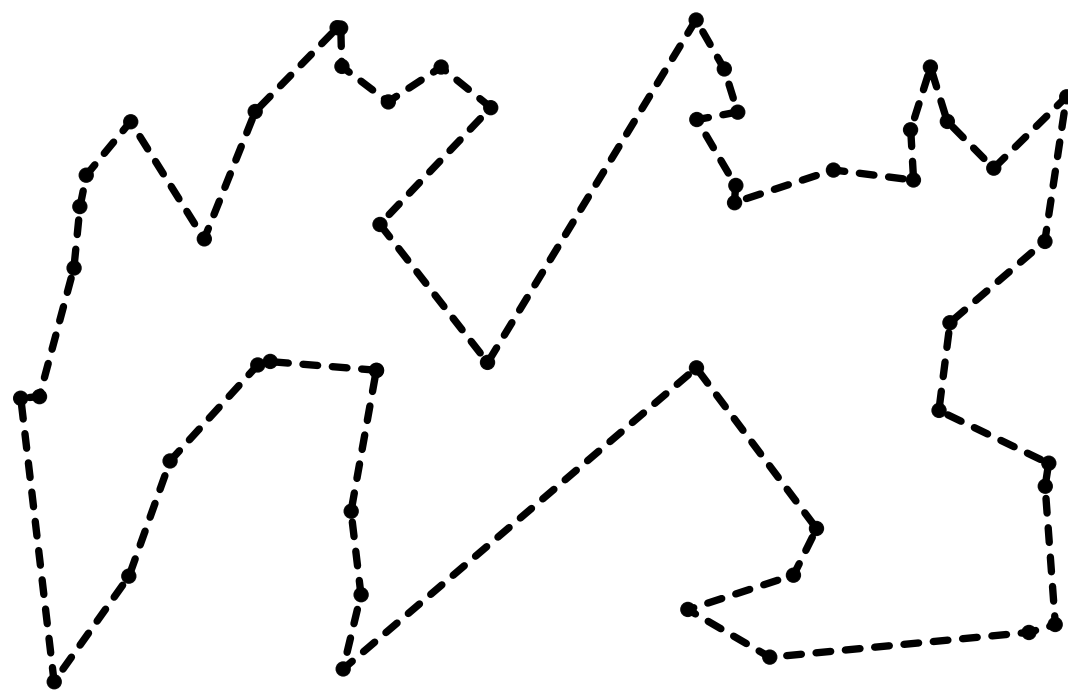
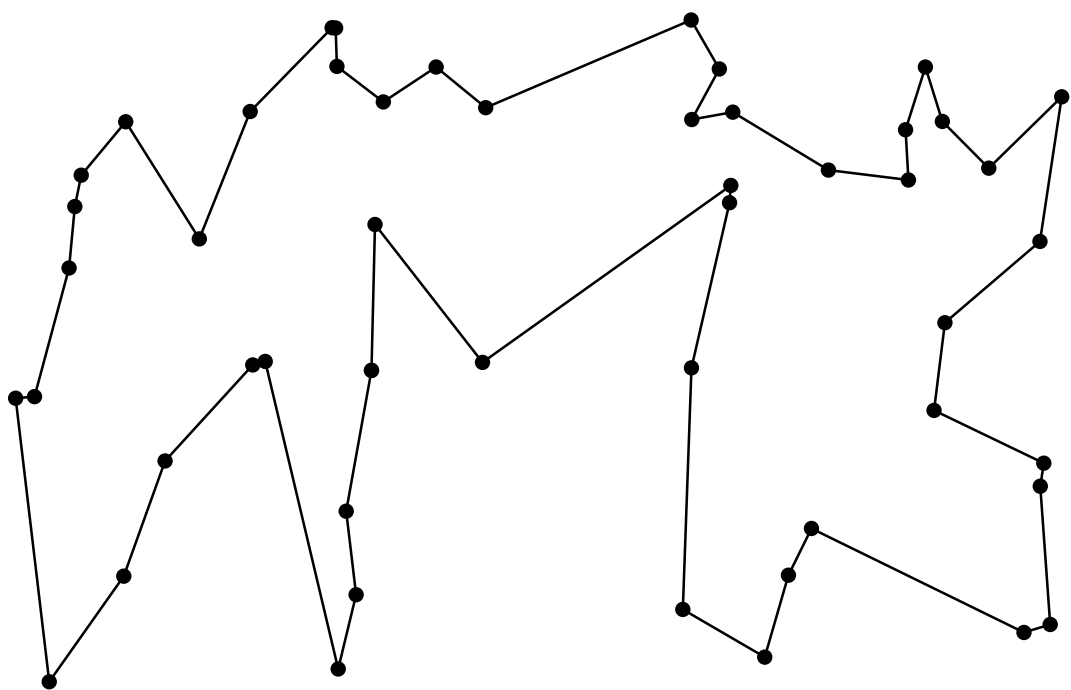
- EAs/MHs are the schemes of algorithms that refer to the representation of solutions using operators
- The characteristics of the problems that make them suitable for use EAs/MHs therefore depend on the operators, not just on the problem itself!
- On the other hand, the question is whether it is possible to define for each problem a set of operators that ensures the presence of the desired characteristics?
  - It is certainly possible to "spoil"

# Distance preserving crossover

- Distance – measure opposite to similarity – number of different features
- Distance preservation = preservation of common features of parents
- The offspring contains all the features common to both parents. The rest is set randomly or heuristically.

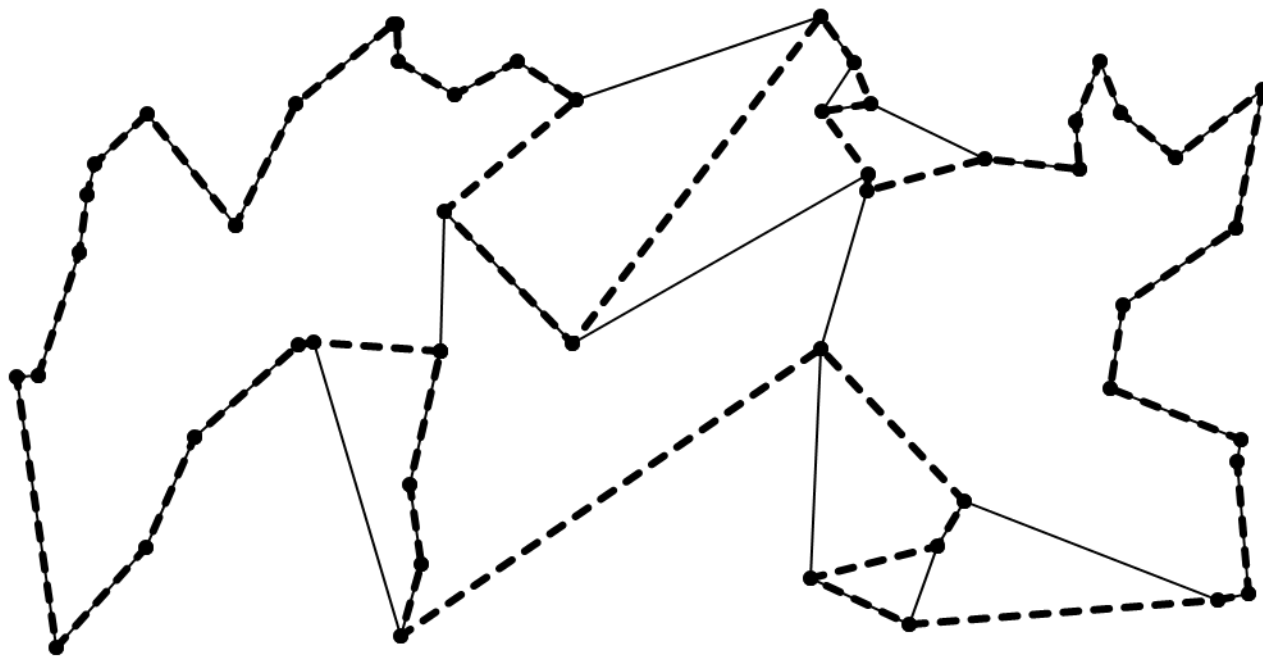


# Two exemplary local optima of a TSP instance

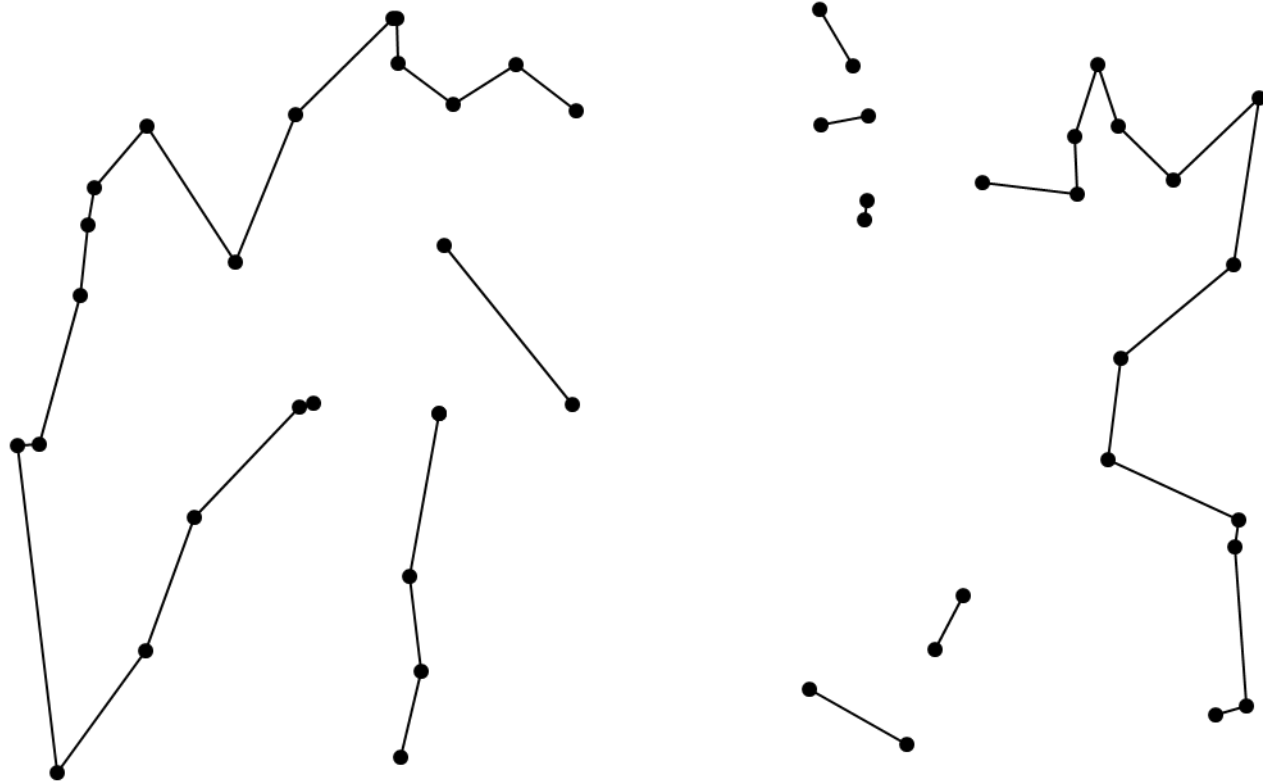




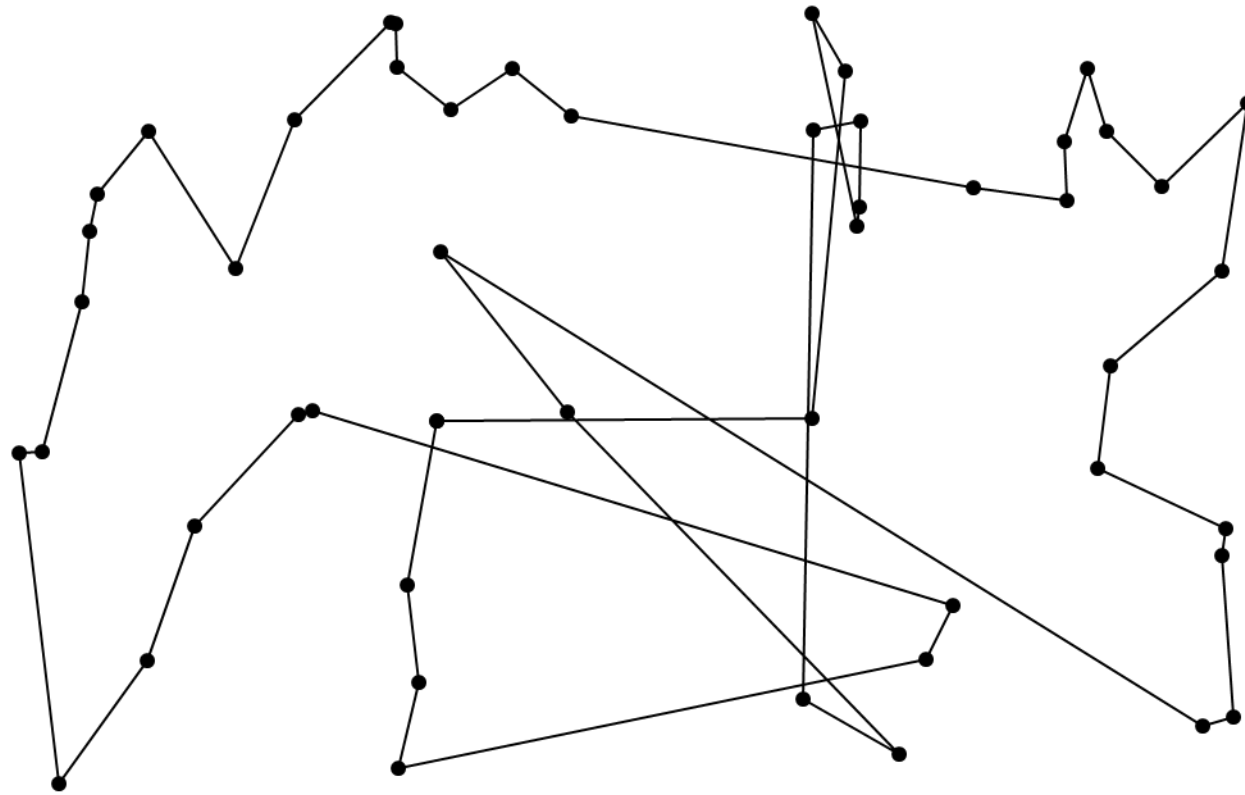
Two exemplary local optima of a TSP instance



# Common edges (and paths)



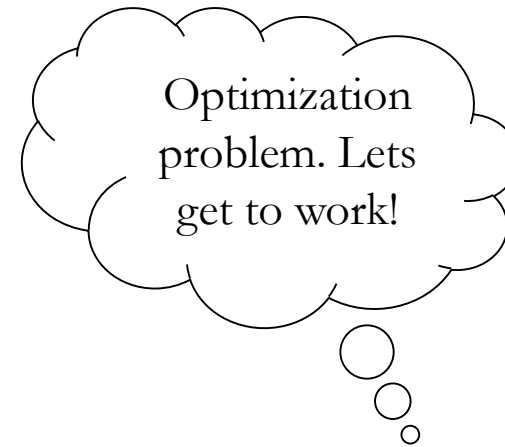
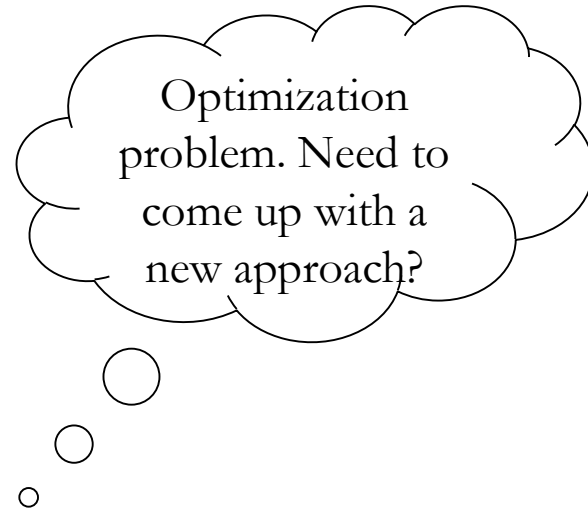
# Random connection of common paths



# Designing EAs/MHs for practical problems

- Getting good results in practice using EAs/MHs is neither obvious nor easy
  - The best EAs/MHs adaptations for classic (simple in definition problems) are based on the results of up to several decades of research (e.g. TSP)
    - Time "rather unacceptable in practice"
  - Algorithms based on EAs/MHs must be competitive with other tools, e.g. mathematical programming solvers

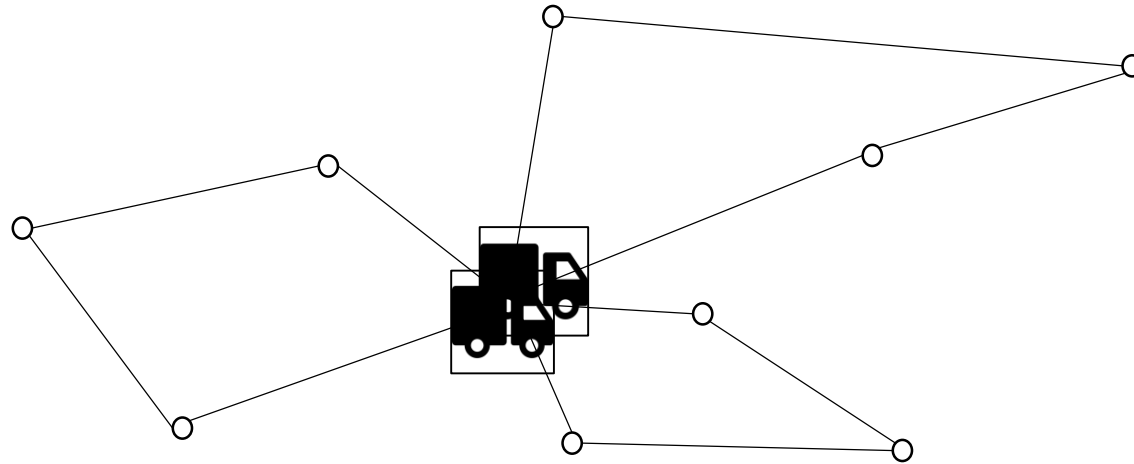
# "Scientific/artistic" vs engineering approach to adapting EAs/MHs to a specific problem



# Systematic/engineering design of effective optimization algorithms

- Construct an effective local search
  - Heuristic construction of good initial solutions
  - Choice of a neighborhood with low autocorrelation rates
  - Use of techniques such as deltas, use of moves evaluations from previous iterations, candidate moves, general techniques to improve code effectiveness
- Formulate hypotheses about the important features of the solutions
- Perform quality-distance correlation tests for different measures of similarity – different features of solutions
- If you observe correlation for certain features, design the recombination/destroy-repair operator(s) using these features and use a hybrid population algorithm
- If there is no correlation, apply multiple start local search

# Example – vehicle routing (VRP) – different versions



# Tested features

- Number of common pairs of arbitrary vertices – assigned in both solutions to a single route
- Number of common edges
- Number of common arcs
- Number of common pairs of vertices not connected by arcs/edges
- Number of common edge pairs
- Number of common ordered pairs of vertices
- Difference in relative positions of vertices in routes



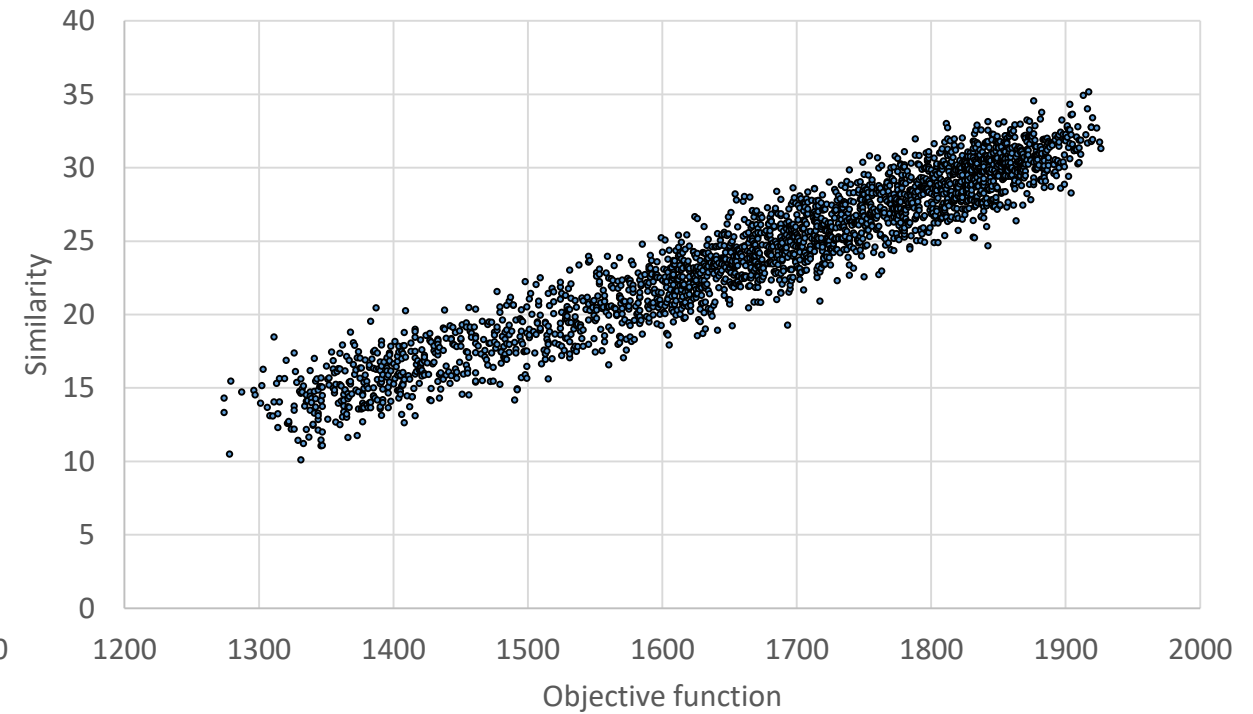
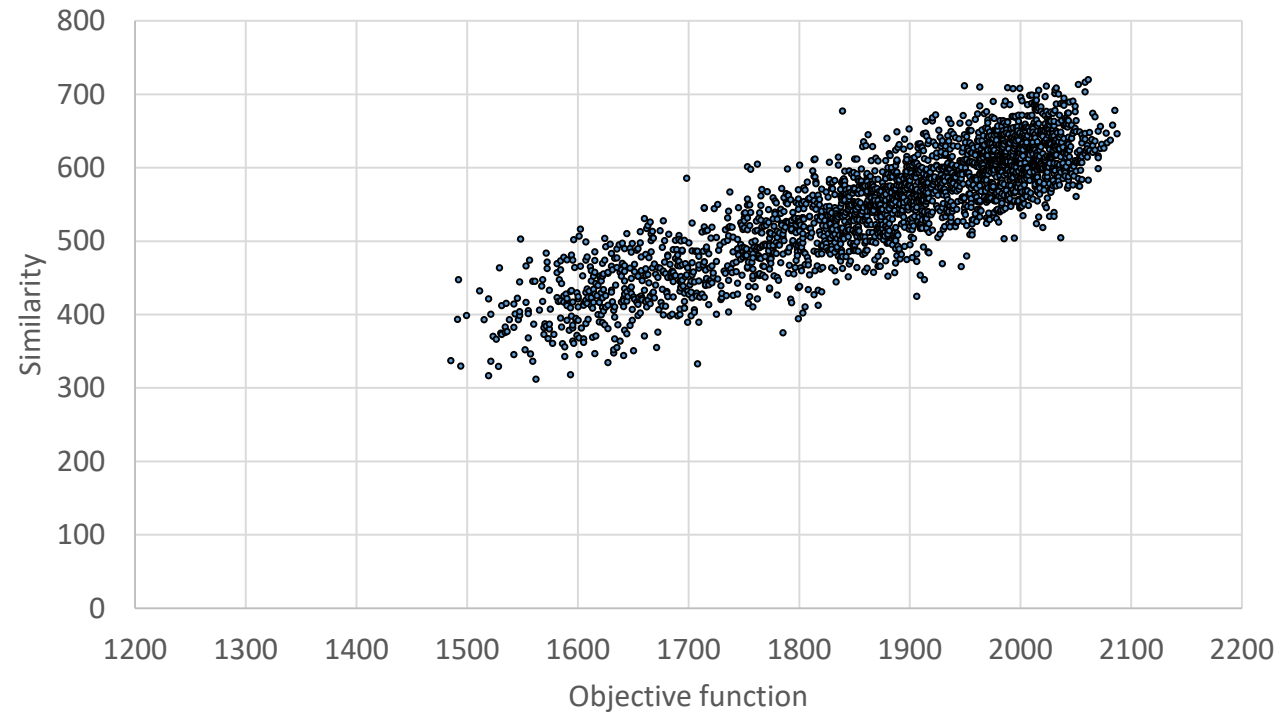
# Correlation sizes

	Measure of similarity						
Instance	1	2	3	4	5	6	7
CVRPTW							
rc_2_4_1	-0.79	-0.4	-0.34	-0.78	-0.67	-0.44	0.02
rc_2_8_1	-0.7	-0.62	-0.59	-0.69	-0.66	-0.56	-0.32
rc_1_4_10	-0.42	-0.21	-0.28	-0.4	-0.28	-0.25	-0.2
rc_1_8_10	-0.33	-0.13	-0.23	-0.33	-0.21	-0.37	-0.2
c_1_10_10	-0.81	-0.65	-0.68	-0.81	-0.74	-0.76	-0.48
rc_2_10_2	-0.79	-0.15	-0.16	-0.79	-0.5	-0.42	-0.13
PDPTW							
lc1_4_4	-0.75	-0.52	-0.56	-0.72	-0.62	-0.4	-0.43
lc2_10_4	-0.51	-0.29	-0.39	-0.5	-0.43	-0.4	-0.34
lr1_6_3	-0.62	-0.6	-0.56	-0.59	-0.55	-0.16	-0.17
lr2_10_8	-0.5	-0.46	-0.39	-0.5	-0.55	-0.28	-0.18
lr2_6_10	-0.49	-0.46	-0.42	-0.49	-0.52	0.05	-0.32
lrc2_10_3	-0.57	-0.36	-0.29	-0.56	-0.35	-0.1	-0.07
lrc2_4_10	-0.52	-0.45	-0.39	-0.52	-0.51	-0.08	-0.26
lrc2_8_10	-0.51	-0.43	-0.38	-0.5	-0.47	-0.02	-0.29
TOPTW							
pr05	0.8	0.92	0.89	0.78	0.87	0.49	0.84
pr10	0.89	0.95	0.93	0.88	0.91	0.6	0.9
pr15	0.83	0.94	0.89	0.81	0.86	0.62	0.85
pr16	0.83	0.93	0.89	0.81	0.87	0.57	0.87
pr19	0.83	0.93	0.88	0.82	0.86	0.51	0.79
pr20	0.86	0.93	0.9	0.84	0.89	0.54	0.81

Minimized objective f.

Mazimized objective f.

# Correlation examples – maximized objective function



# Route-based recombination for VRP

Offspring solution  $O \leftarrow$  empty solution

Select at random parent  $p$

**repeat**

    Choose from  $p$  the route  $r$  with the largest number of common, not yet allocated vertices

    Remove from  $r$  vertices that have already been added to  $O$

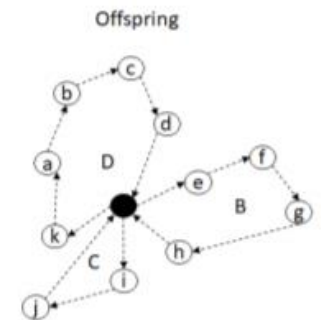
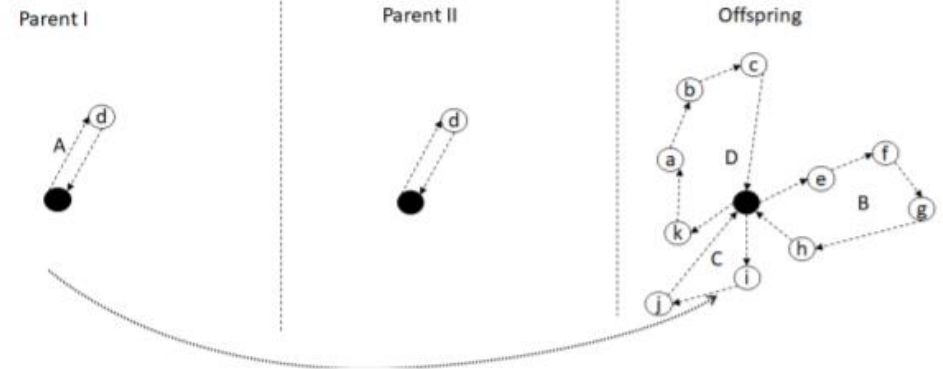
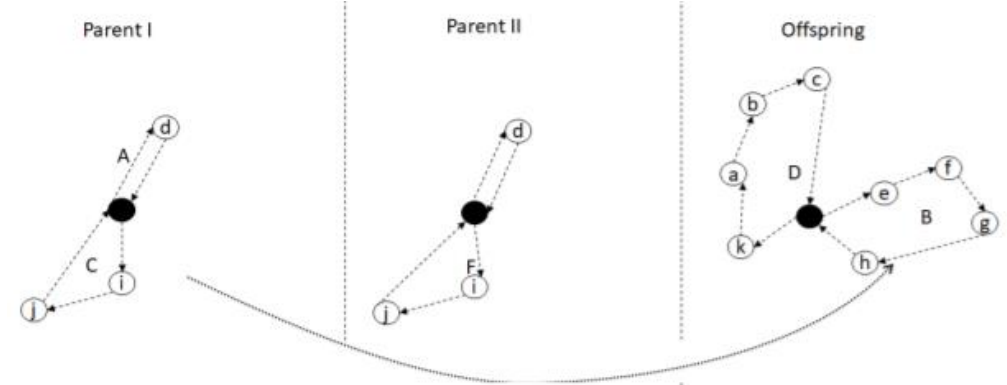
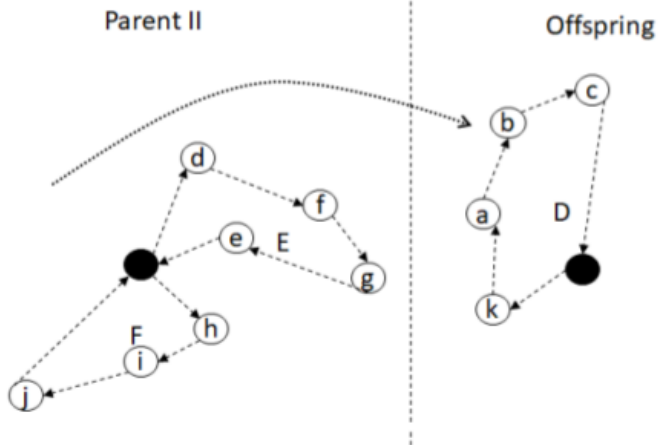
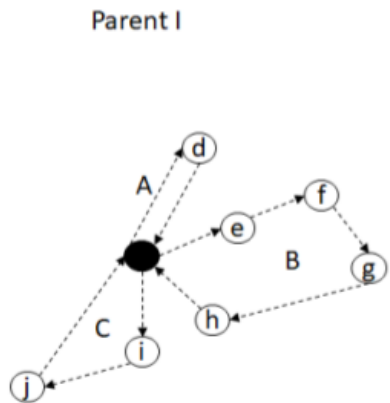
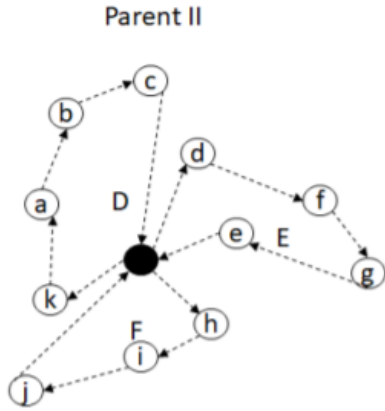
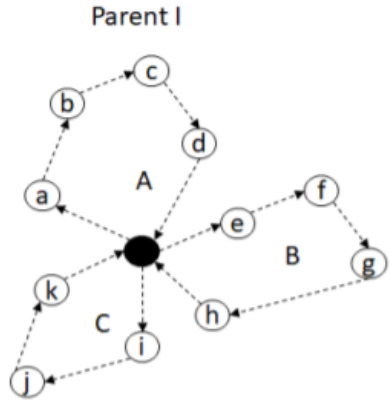
    Add route  $r$  to  $O$

**until** the number of routes in the  $O$  child is equal to the number of routes in  $p$

Lock all common arcs and edges in the offspring

Insert unallocated vertices using a greedy procedure and return  $O$

# Route-based recombination for VRP



# Another example for VRP - Modified Selective Route Exchange Crossover

- Treat the routes of both parents as sets of vertices
- Define the problem of maximum coverage – selection of a set of routes (of size such as in one of the parents) with a lexicographic criterion:
  - Maximization of covered vertices, then minimization the sum of the lengths of the selected routes
- Solve the problem of maximum coverage with a simple hybrid evolutionary algorithm with a local and global list of tabu (prohibited) solutions (HAE inside recombination)
- For redundantly allocated vertices (to two routes), randomly select one of them (i.e. remove from the other)
- Insert unallocated vertices using a greedy procedure

# Order-based recombination

Copy parent A to the offspring

Find the relative positions of the vertices in parents A and B

Find routes in A that contain at least one pair of vertices with relative positions different than in B

Sort these routes by relative positions in B

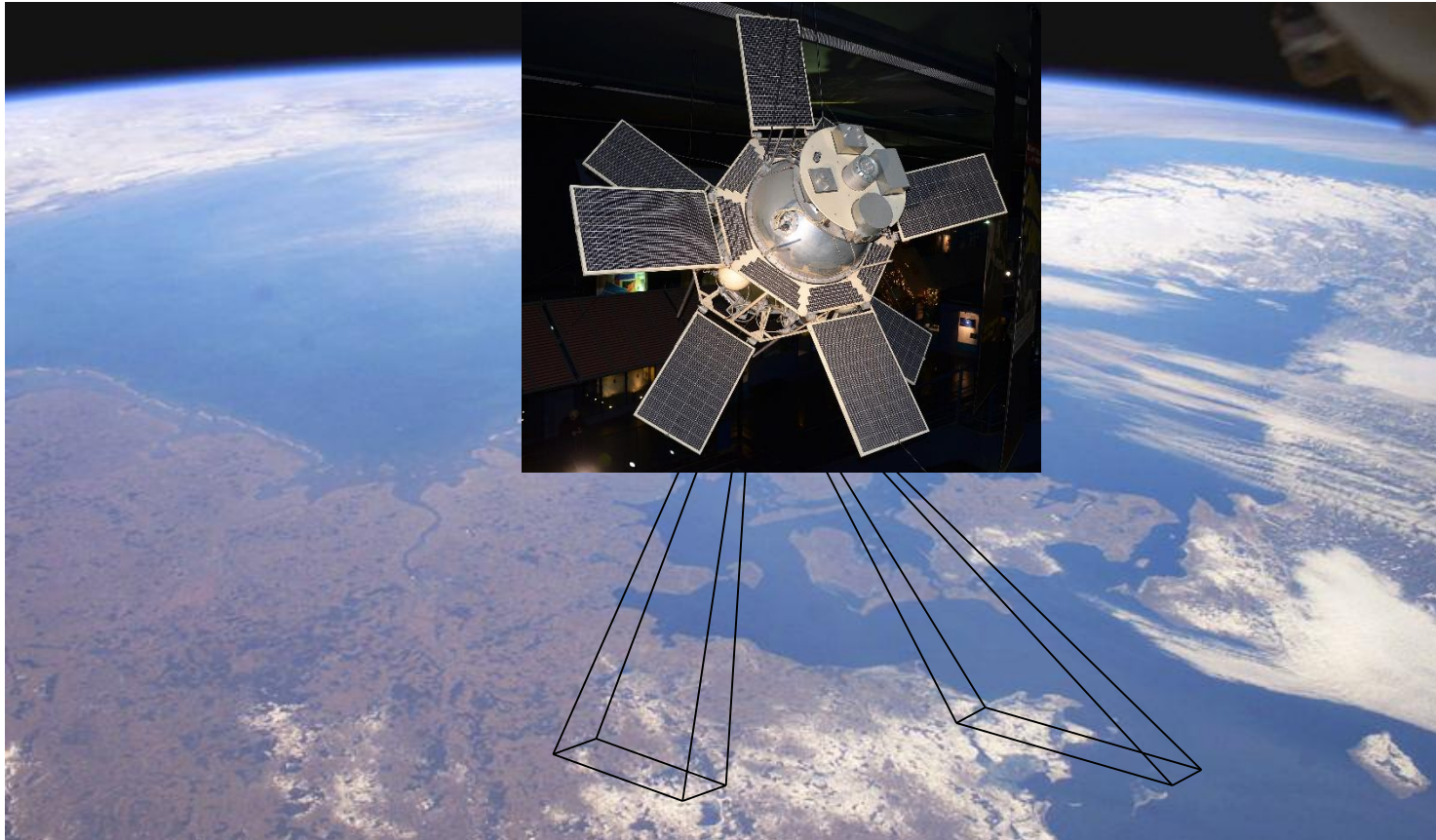
For each combination of 1, 2 or 3 routes

If such a combination has not yet been considered, replace these routes in the descendant

If you all of the above combinations have been tested

Perform a random number of inversions of vertex pairs

# Case study – management of observation satellites



# Problem description

- Orders – collections of photos
- Regular photos – one shot required
- 3D photos – two shots required
- A given photo can be taken in two ways (shots) depending on the direction of movement of the camera
- The profit from the execution of an order depends in a non-linear manner on the executed area of the order
- For each shot, there is an earliest and latest start time resulting from the satellite's field of view – the time window
- Each shot has an execution time
- Switching between shots takes non-negligible time depending on their position
- **Goal: Select photos and shots and their schedule maximizing profit**



# Analogies with other problems

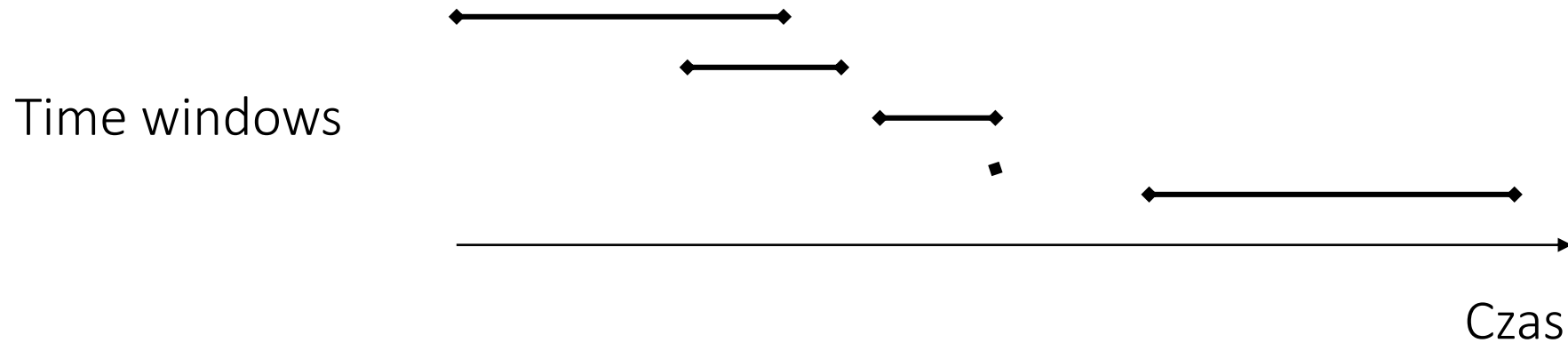
- Knapsack problem – photo selection
  - More complicated constraints and calculation of objective function
- Task scheduling – scheduling tasks on one machine with switching times and time windows
  - The (direct) goal is not to minimize completion time
  - In scheduling, usually, all tasks must be completed, here most are not
- TSP – tasks are cities, switching times are travel times between cities
  - Not all tasks need to be completed, most are not
  - Execution Times need to be considered
- VRP – tasks are cities, switching times are travel times between cities, shooting times are loading/unloading times, loading/unloading time windows
  - Not all tasks need to be completed, most are not

# Analogies with other problems

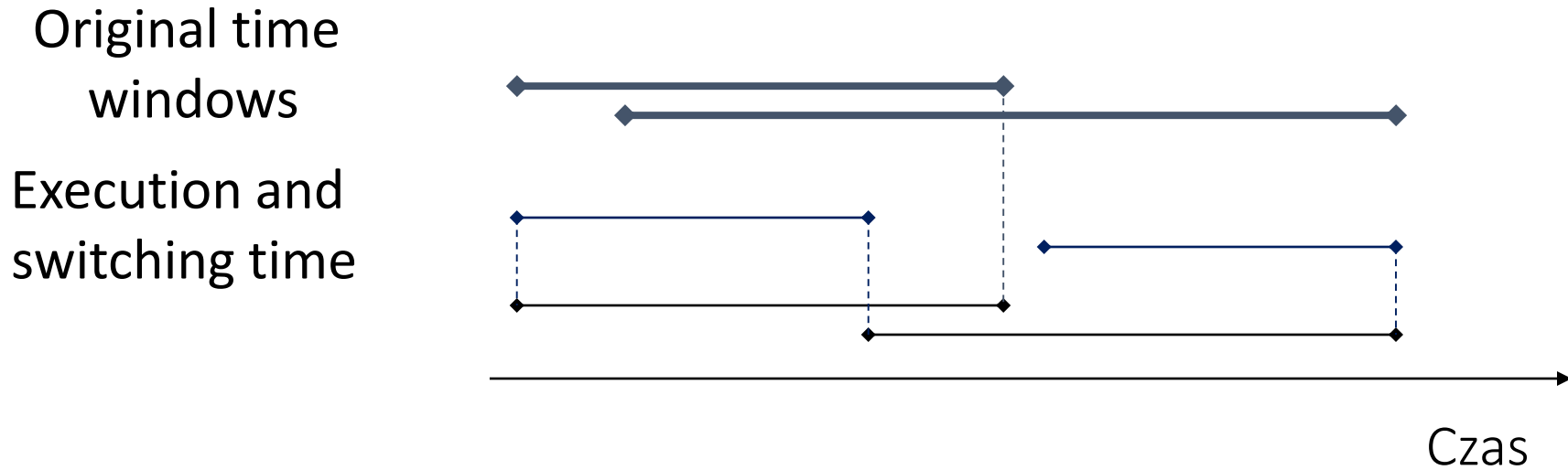
- Trip planning problem - tasks are visited places, switching times are travel times, execution times are times of stay, time windows – opening times of attractions
  - Few works on this problem

# Solution representation

- An ordered list (sequence) of selected shots
- The start time windows follow from a sequence



# Propagation of time windows



# Initial solution construction

- Greedy heuristics motivated by heuristics for the knapsack problem
- One shot is inserted in a given step
- The shot and insertion place with the best ratio of profit growth to time taken is selected
- Time taken includes execution and switching times
- Approximate (linear) profit calculation
- Time windows reduce the scope of the search for insertion position
- Randomization - a certain number of initial shots are inserted randomly

# Local search

- Move – inserting a shot in the best, i.e. giving the largest increase in profit, place combined with the removal of the necessary shots
- Profit is calculated exactly
- Time windows reduce the scope of the search for insertion position

# The problem is difficult!

- Basic local search extensions – tabu search and iterative local search do not give noticeable improvements in results
  - ... or weak neighborhood

# Hypotheses about important features

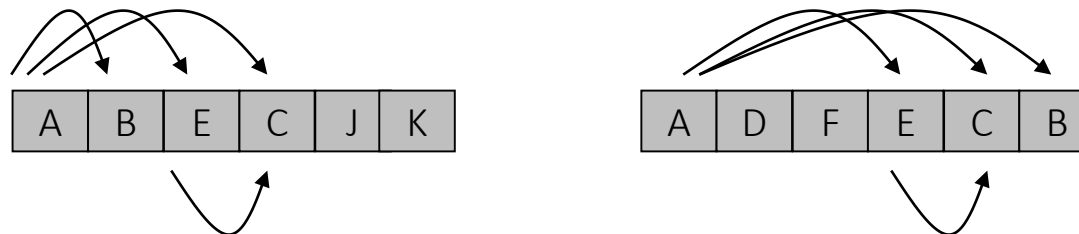
- Selected shots



- Orders selected for execution – completed area



- Order of pairs of shots



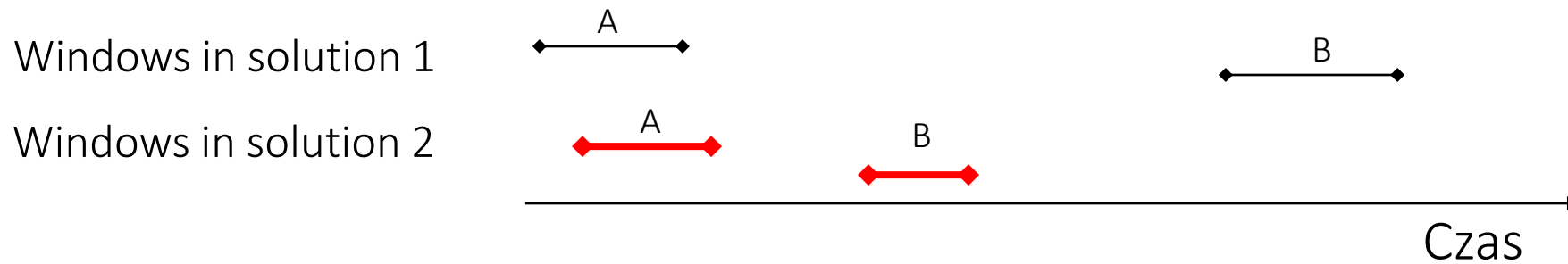


# Hypotheses about important features

- Pairs of adjacent shots – the equivalent of arcs in TSP

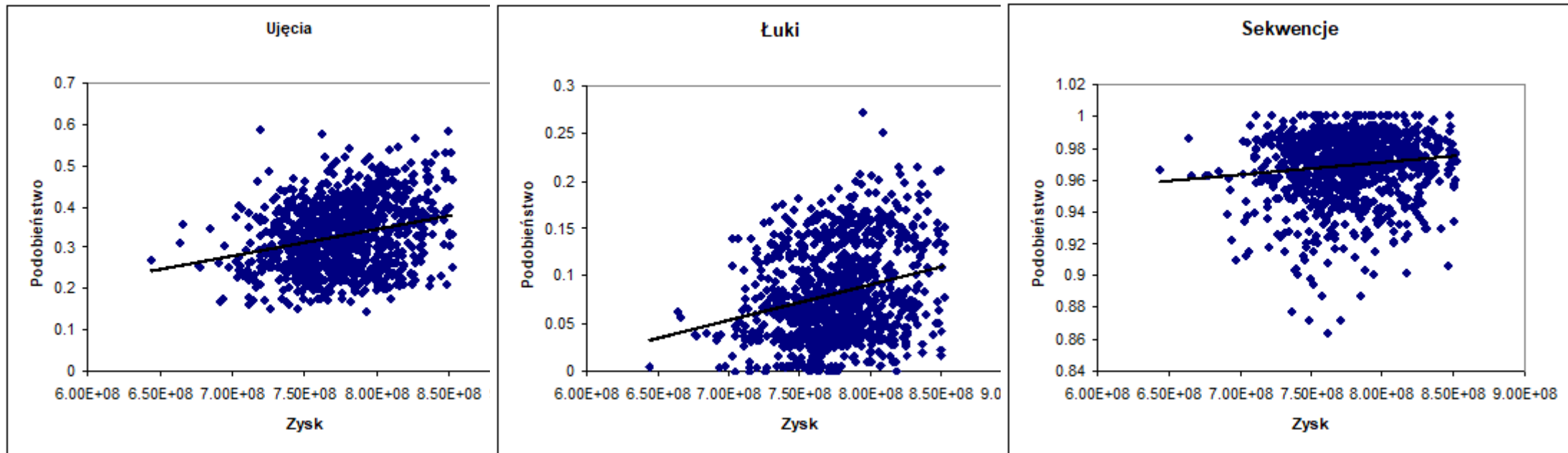


- Start time windows – distance between central points



# Quality-distance correlations

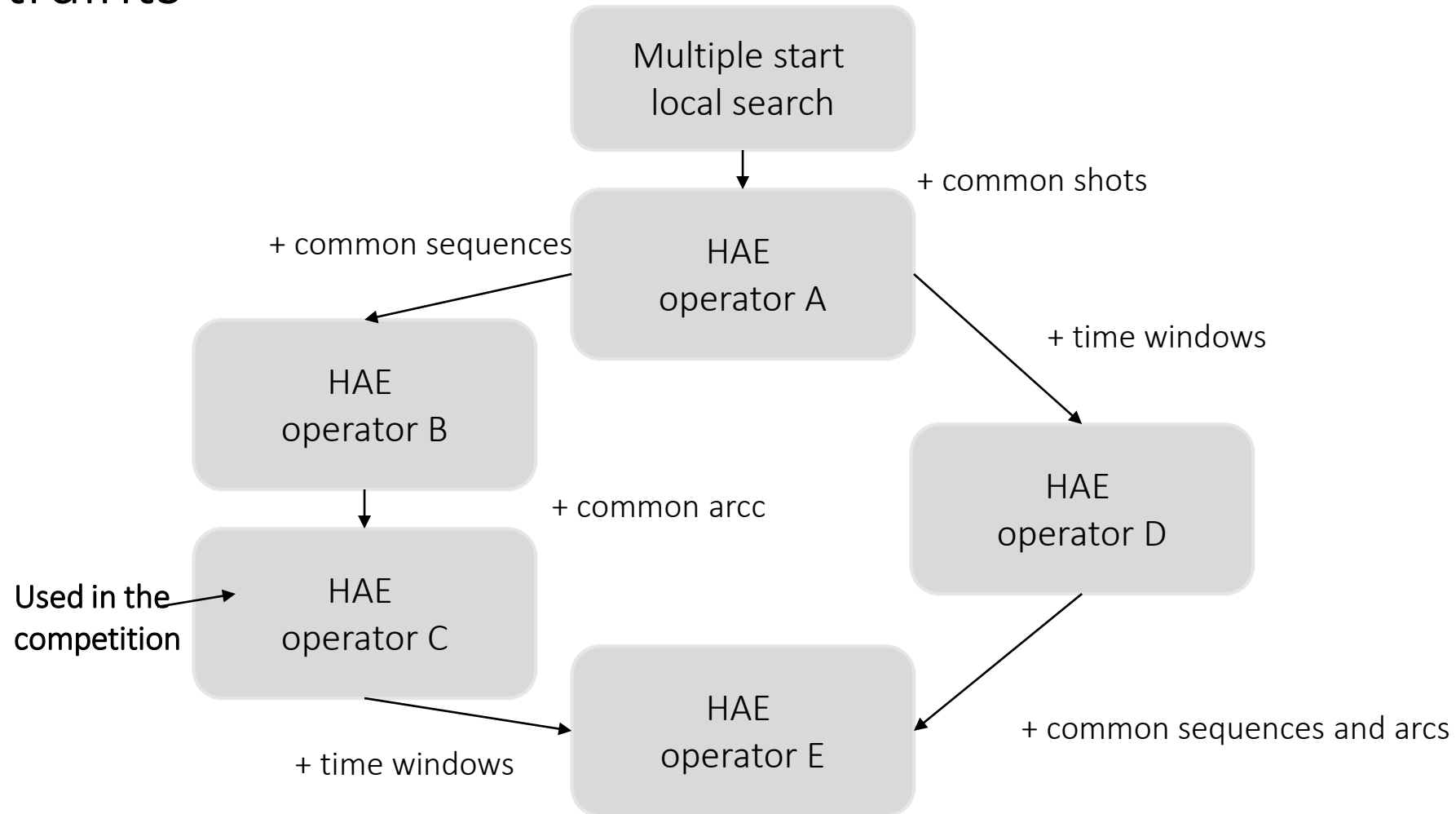
- For a given solution, the average distance from all **better** solutions



# Conclusions

- The recombination operator should maintain a common
  - Captures (ensures preservation of photos and orders)
  - Sequences
  - Arcs
  - Time windows (i.e. limit them to reanges similar as in parents)

# Series of recombination operators based on the construction heuristic for initial solutions with additional constraints



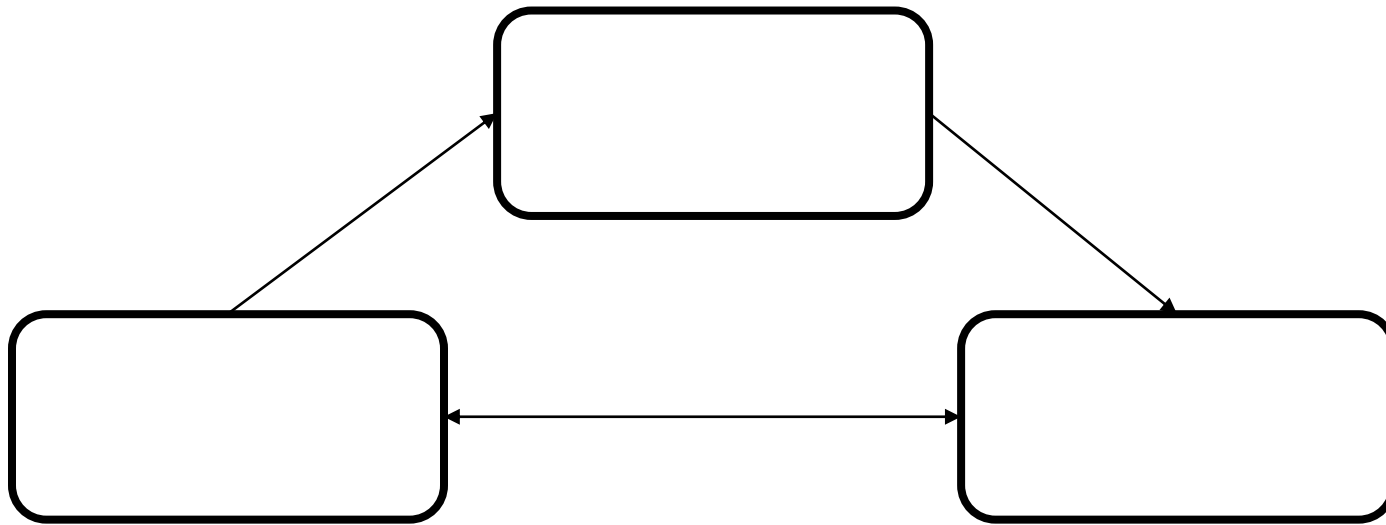
# Recombination operator and hybrid evolutionary algorithm

- The solution, after preservation of common features, is supplemented by the insertion of shots heuristics
- Local search is run with a probability of 2.5%
- Common arcs may not be broken
- Algorithm with full elitism
- Observations:
  - Significant improvement in results over multiple start local search despite the low probability of improvement of the solution in a given step
  - Long calculation time for large instances
  - Large dispersion of results for large instances

# Improvements of local search

- Limited number of evaluated moves
- Insertions of all shots that occurred in at least one parent are considered – candidate moves
- The insertion of the remaining shots is considered with a probability of 10%
- Observations
  - Satisfactory calculation time
  - Large dispersion of results for large instances
  - Only very slight improvement in result dispersion when increasing population size


# Population island model



- Observations
  - Satisfactory calculation time
  - Lower dispersion of results in the absence of improvement in the best results – better average results

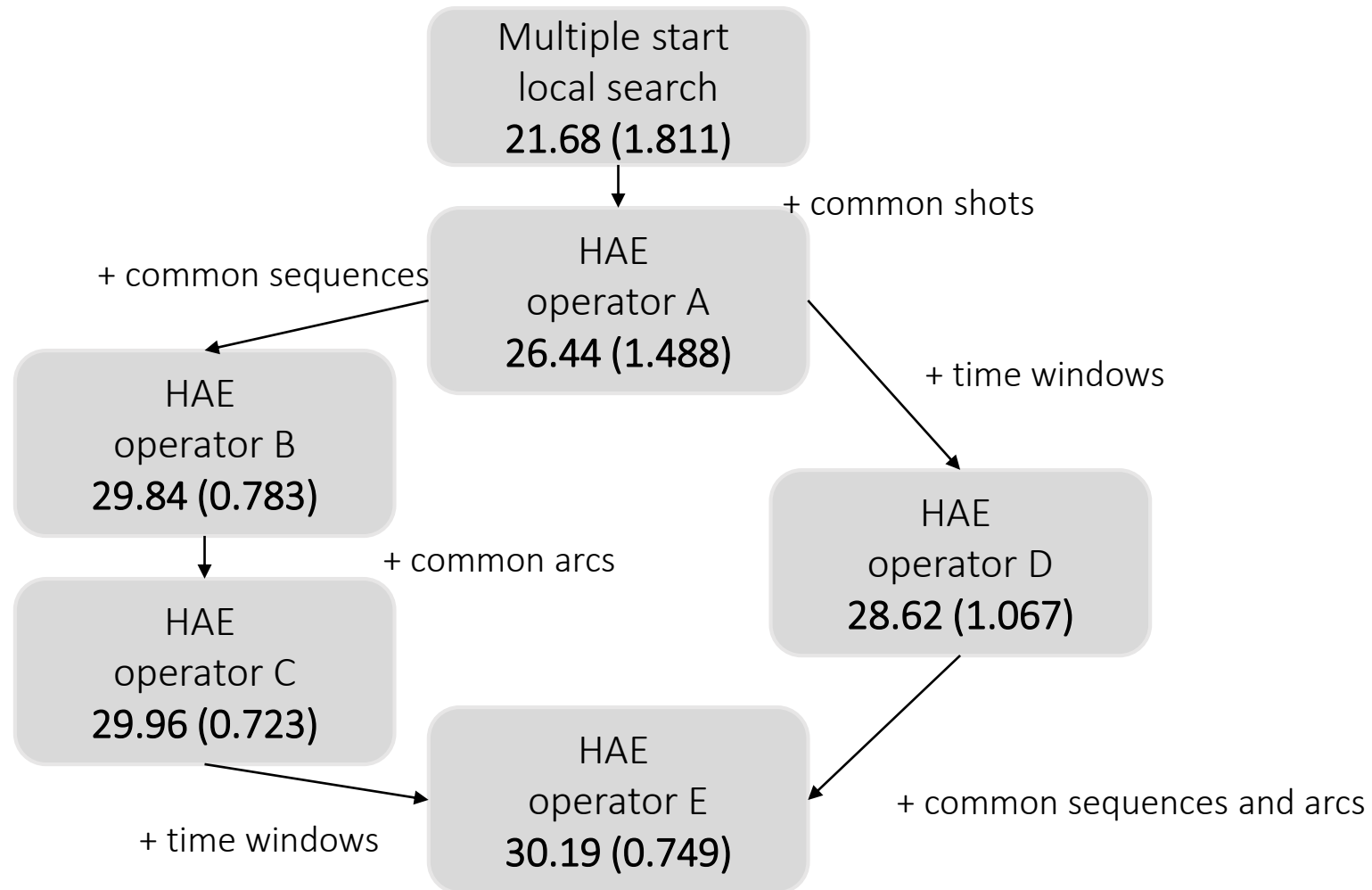
# Results of the competition – operator C

Team	Evaluation
1 (best)	31.76
2	30.84
3	30.28
4	30.1
5 (Author)	29.91
6	29.88
7	29.84
8	29.69
9	29.22
10	29.08
11	23.56

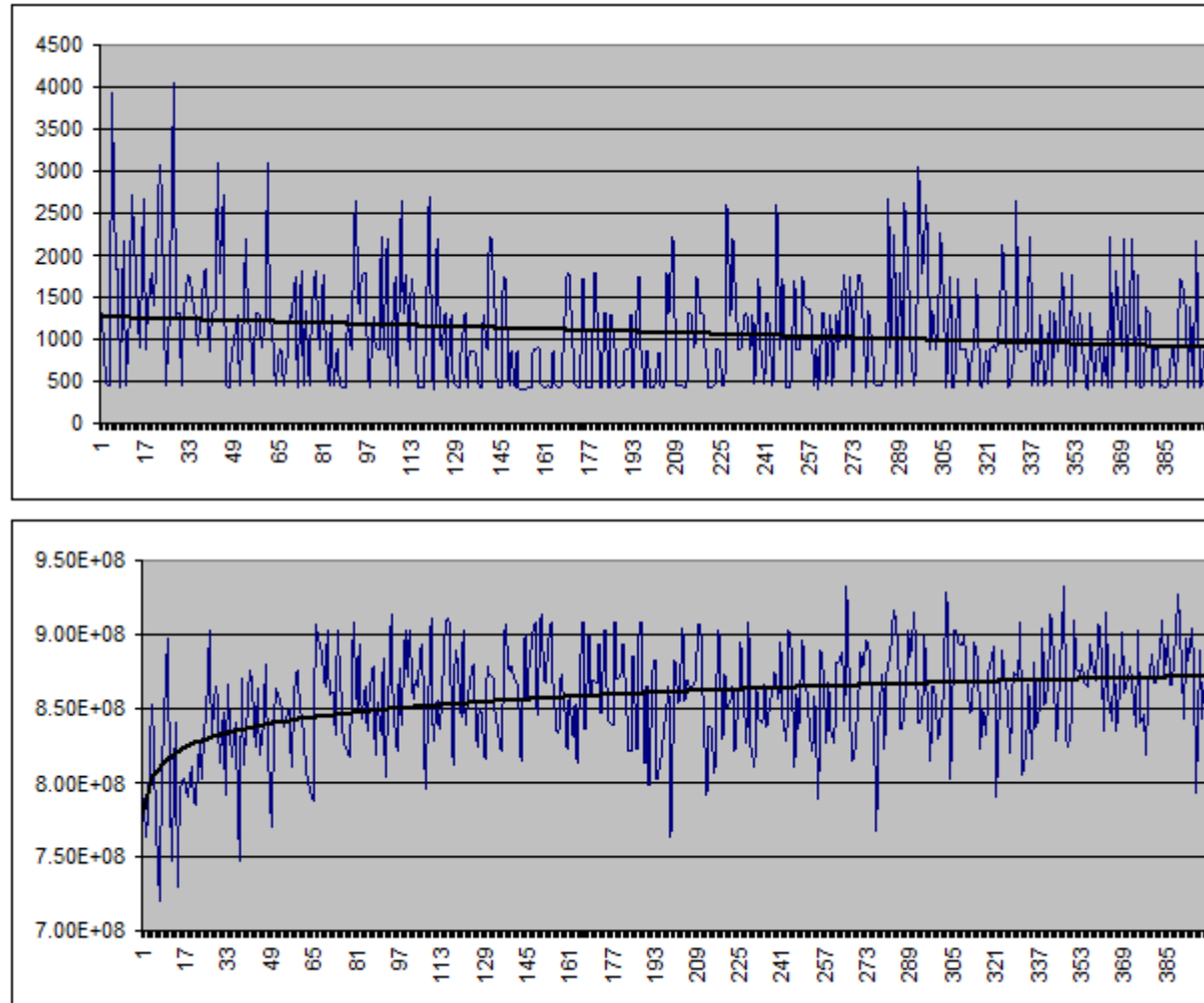




# Comparison of operators



# Time and quality of local search results

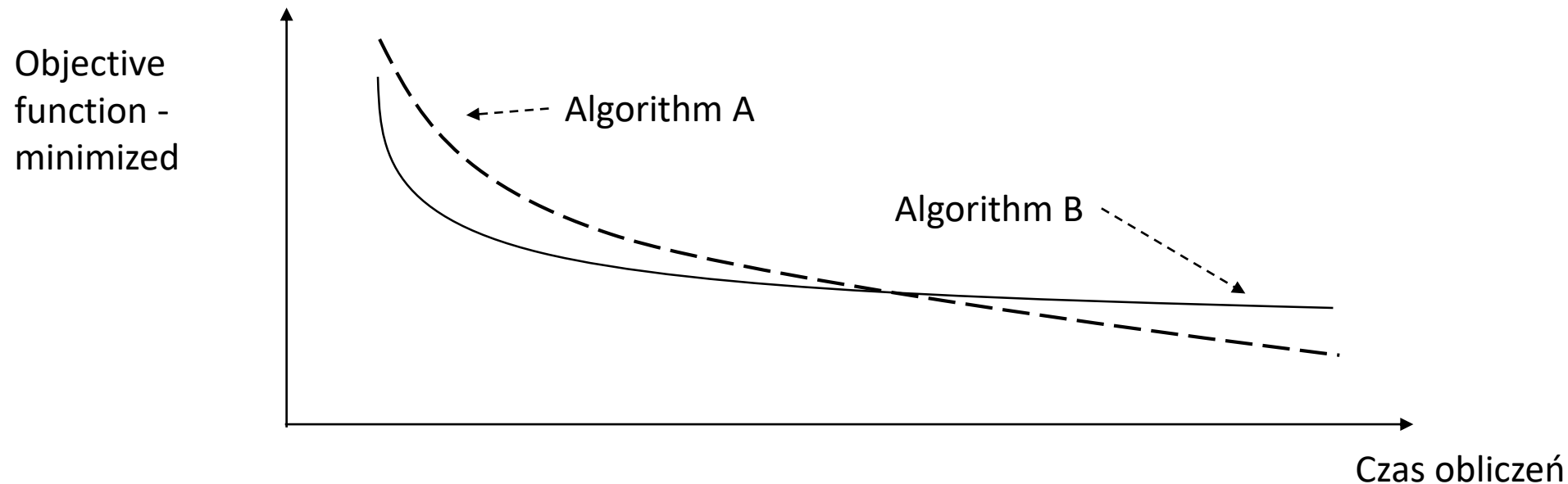


# Summary – a systematic approach to EAs/MHs design

- Neighborhood, recombination, destroy-repair operators can be designed systematically, not by trial and error
  - This means less, shorter and easier to implement computational experiments
  - In addition, we understand why our algorithm works (or does not work)

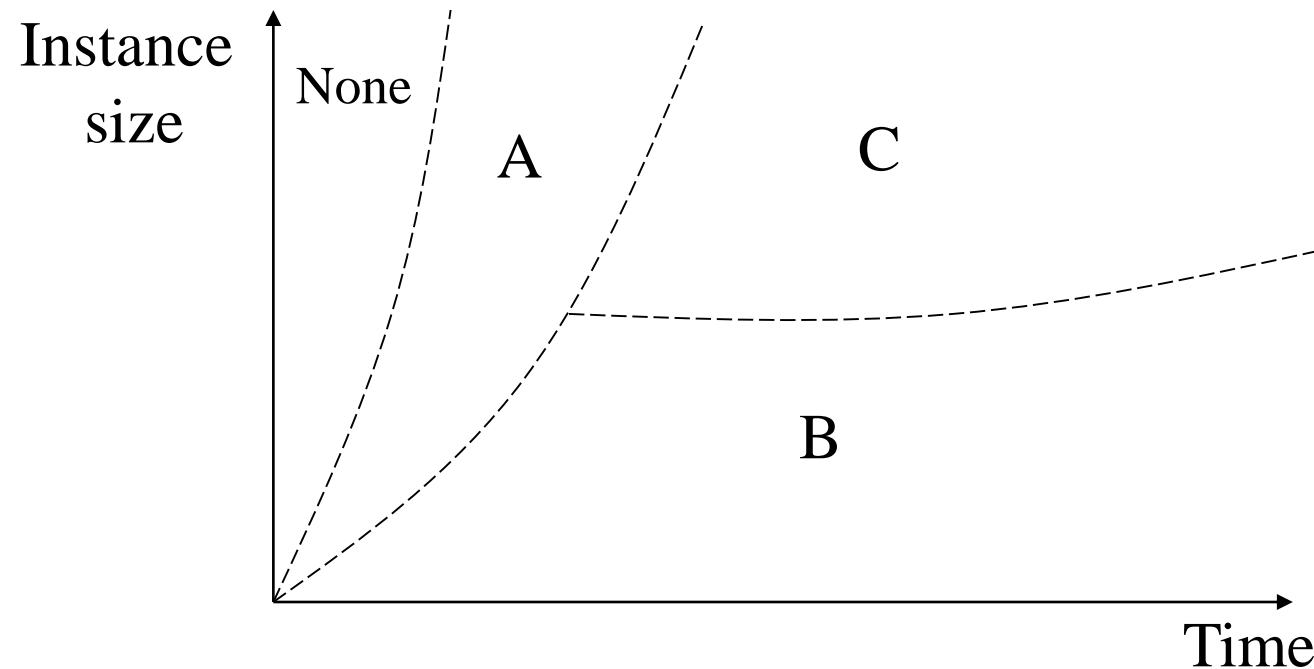
# Experimental evaluation of heuristic algorithms

- Two main criteria:
  - Quality – the value of the objective function
  - Calculation time
- Relative quality may vary depending on the available calculation time



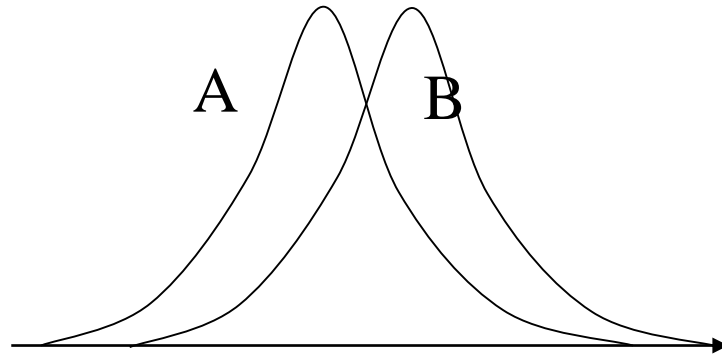
# Ranking on the time-instance size surface

- The best algorithm with A, B and C for a given time and size

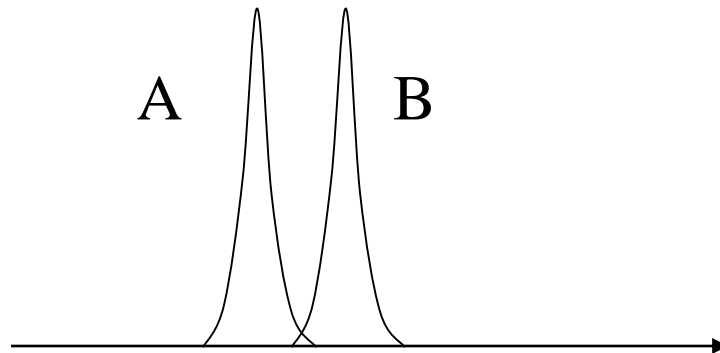


The relative dispersion of results (values of the objective function) often decreases as the size of the instance increases

- Algorithm comparison becomes clearer



Small instance



Large instance

Exemplary trends – new interesting ideas

# Matheuristics - Use of EAs/MHs in solvers

- Mathematical programming solvers - development in recent years
  - Preprocessing – removal of redundant (always inactive) constraints and variables
  - Branch and price – combination of branch and bound with column generation
  - Branch and cut – combination of branch and bound with cutting planes
- EAs/MHs may be used for:
  - Determining the order of branch selection
  - Heuristic search for inactive variables and constraints
  - Heuristic column generation and cutting planes



# Matheuristics – the use of solvers in EAs/MHs

- E.g. recombination operators using accurate solvers – e.g. choosing routes for a VRP problem from a larger set of routes (set covering problem), e.g. coming from many solutions
- Probe Method

# AI, ML, NN in AEs/MHs

- E.g. design of operators/heuristic with NN
  - Dynamic Partial Removal: A Neural Network Heuristic for Large Neighborhood Search, Mingxiang Chen, Lei Gao, Qichang Chen, Zhixin Liu, arXiv:2005.09330
- Autoencoders as perturbation/recombination operators
- Direct solution generation, instance -> solution with NN
  - Lu Duan, Yang Zhan, Haoyuan Hu, Yu Gong, Jiangwen Wei, Xiaodong Zhang, and Yinghui Xu. 2020. Efficiently Solving the Practical Vehicle Routing Problem: A Novel Joint Learning Approach. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20). 3054–3063.
- Review works
  - Analytics and Machine Learning in Vehicle Routing Research, Ruibin Bai et al., arXiv:2102.10012.
  - Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. European Journal of Operational Research, 290(2):405–421, 2021.
  - Quentin Cappart, Didier Chételat, Elias Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks, 2021, arXiv:2102.09544.

# Quantum computing

- Quantum adiabatic algorithm (QAA) (quantum annealing) on noisy intermediate-scale quantum (NISQ) devices
- Quantum approximate optimization algorithm (QAOA)
- Variational quantum algorithms (VQA) – quantum-classical

**Science** Current Issue First release papers Archive About Submit man

RESEARCH ARTICLE f t in d w e

## Quantum optimization of maximum independent set using Rydberg atom arrays

[S. EBADI](#), [A. KEESLING](#), [M. CAIN](#), [T. T. WANG](#), [H. LEVINE](#), [D. BLUVSTEIN](#), [G. SEMEGHINI](#), [A. OMRAN](#), [J.-G. LIU](#), [...] [M. D. LUKIN](#) +15 authors [Authors Info & Affiliations](#)

SCIENCE • 5 May 2022 • First Release • DOI:10.1126/science.abc6587

131 🔔 🔖 🗣️ 🔍

### Abstract

Realizing quantum speedup for practically relevant, computationally hard problems is a central challenge in quantum information science. Using Rydberg atom arrays with up to 289 qubits in two spatial dimensions, we experimentally investigate quantum algorithms for solving the Maximum Independent Set problem. We use a hardware-efficient encoding associated with Rydberg blockade, realize closed-loop optimization to test several variational algorithms, and subsequently apply them to systematically explore a class of graphs with programmable connectivity. We find the problem hardness is controlled by the solution degeneracy and number of local minima, and experimentally benchmark the quantum algorithm's performance against classical simulated annealing. On the hardest graphs, we observe a superlinear quantum speedup in finding exact solutions in the deep circuit regime and analyze its origins.

🔍 📈 👁️ 🔗 🖨️ 📅 🔖