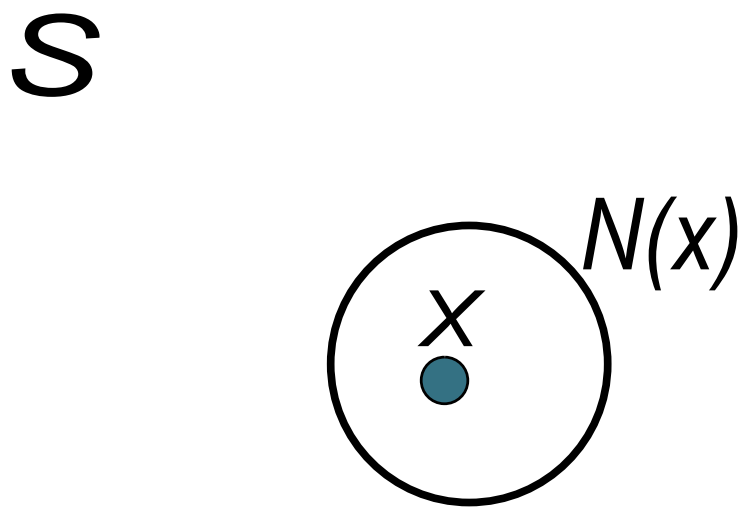
Optimization. Local search

Maciej Komosinski

Institute of Computing Science
Poznan University of Technology
www.cs.put.poznan.pl/mkomosinski

The concept of neighborhood



The definition of neighborhood

- $x \in S$
- a set $N(x) \subseteq S$ of solutions that are "close to" a solution x
- a distance function

$$dist : S \times S \to \mathbb{R}$$

- neighborhood

$$N(x) = \{y \in S : dist(x, y) \leq \varepsilon\}$$

The definition of neighborhood

- $x \in S$
- a set $N(x) \subseteq S$ of solutions that are "close to" a solution x
- a distance function

$$dist : S \times S \to \mathbb{R}$$

- neighborhood

$$N(x) = \{y \in S : dist(x, y) \leq \varepsilon\}$$

- each solution $y \in N(x)$ is called a neighbor of x
- we assume that $y \in N(x) \Leftrightarrow x \in N(y)$

Properties of neighborhood

- size limits
  - for each x, its $N(x)$ contains at least one solution y different from x
  - $N(x)$ should not cover the entire space of acceptable solutions (should not be exhaustive)
- similarity of neighbors
  - $y \in N(x)$ should not be very different from x, so that moving from x to y should not require constructing the new solution y "from scratch"
- "equality"
  - regardless of the choice of the initial solution, any solution belonging to S should be reachable

Examples of neighborhoods for permutation of n items

- k-swap, k-opt
  $N(x)$ – a set of solutions created by removing k items and inserting them in another order
- 2-swap with position preserved

  1—2—3—4—5—6—7—8—9

  1—2—8—4—5—6—7—3—9

  $|N_{2P}(x)| =$

Examples of neighborhoods for permutation of n items

- k-swap, k-opt
  $N(x)$ – a set of solutions created by removing k items and inserting them in another order
- 2-swap with position preserved

  1—2—3—4—5—6—7—8—9

  1—2—8—4—5—6—7—3—9

  $|N_{2P}(x)| = \frac{n(n-1)}{2}$

- 3-swap with position preserved. $|N_{3P}(x)| = ...$

Examples of neighborhoods for permutation of n items

- k-swap, k-opt
  $N(x)$ – a set of solutions created by removing k items and inserting them in another order
- 2-swap with position preserved

  1—2—3—4—5—6—7—8—9

  1—2—8—4—5—6—7—3—9

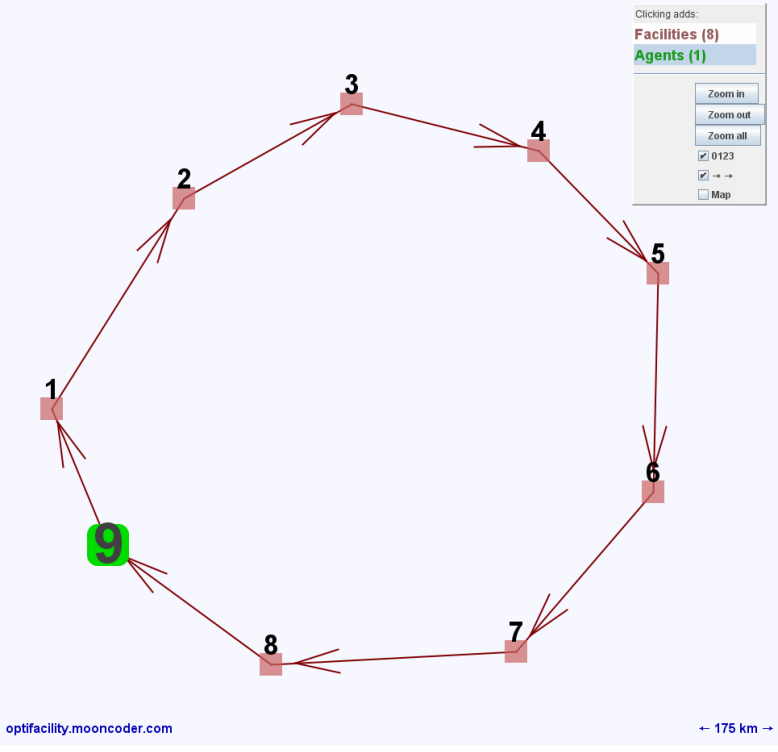
  $|N_{2P}(x)| = \frac{n(n-1)}{2}$

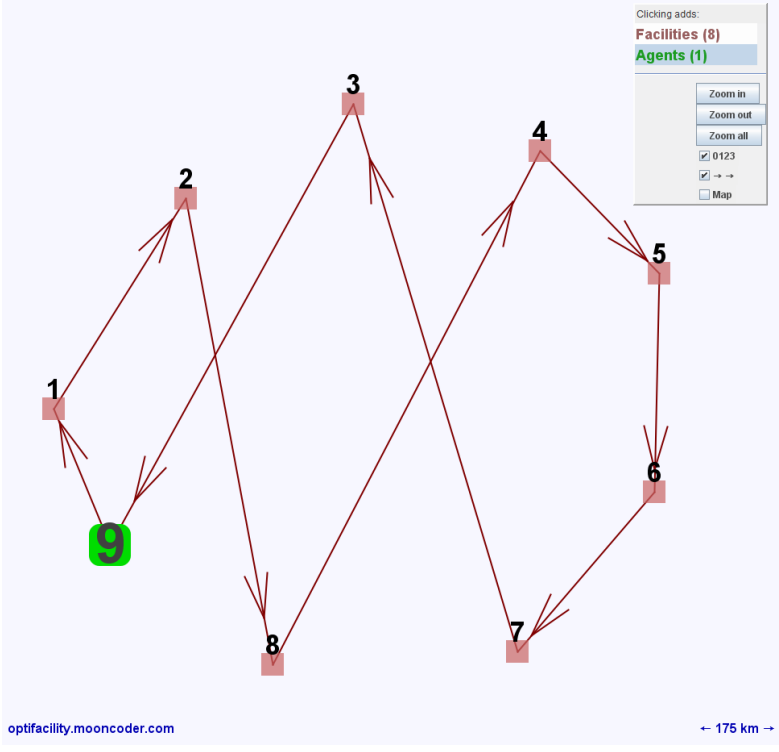
- 3-swap with position preserved. $|N_{3P}(x)| = ...$
- would swapping only adjacent elements constitute a good neighborhood?
- ...

Neighborhood in TSP (swapping cities)

N_{2P}, indexes (3, 8):

$$1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9$$
$$1 - 2 - 8 - 4 - 5 - 6 - 7 - 3 - 9$$

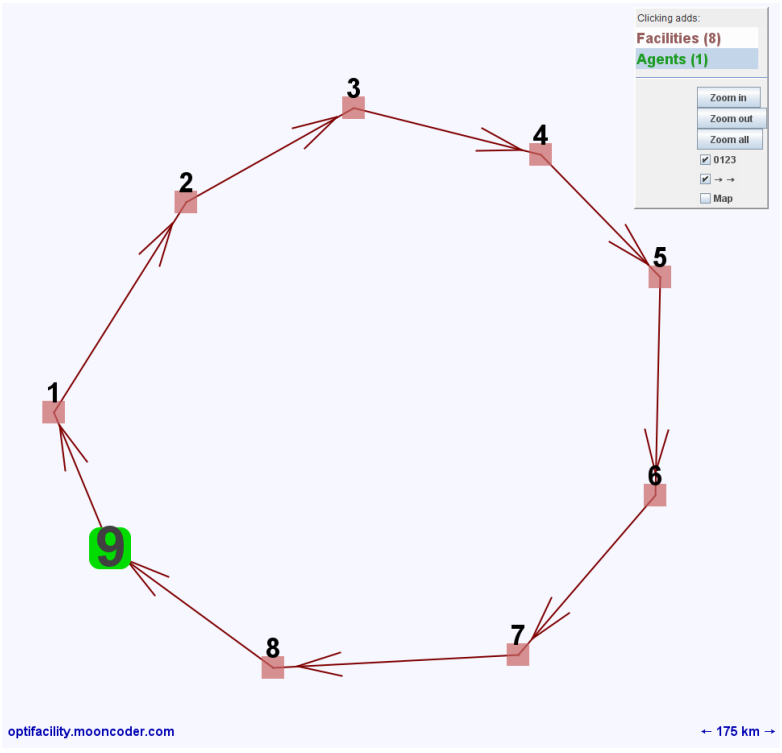


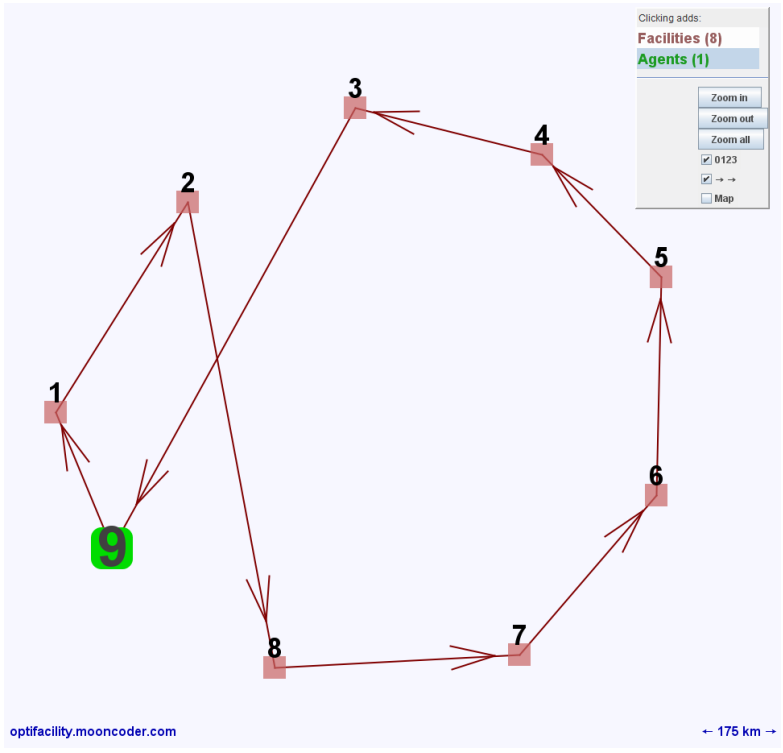


Neighborhood in TSP (path reversal)

N_{2R}, indexes (3, 8):

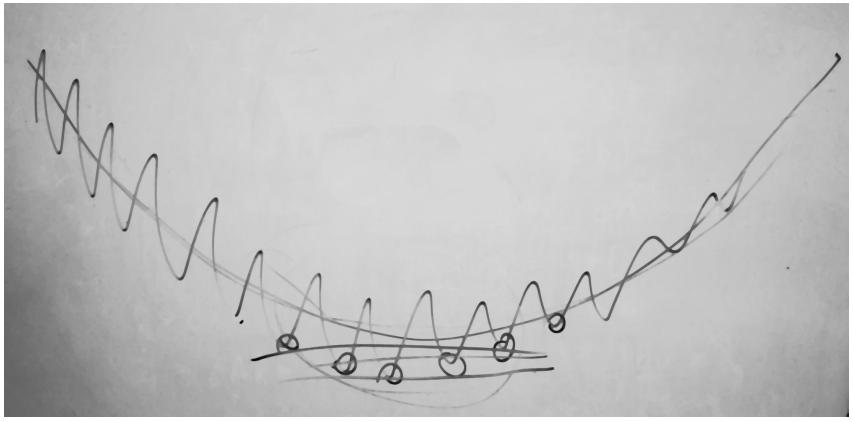
$$1 - 2 - \overrightarrow{3 - 4 - 5 - 6 - 7 - 8} - 9$$
$$1 - 2 - \overleftarrow{8 - 7 - 6 - 5 - 4 - 3} - 9$$





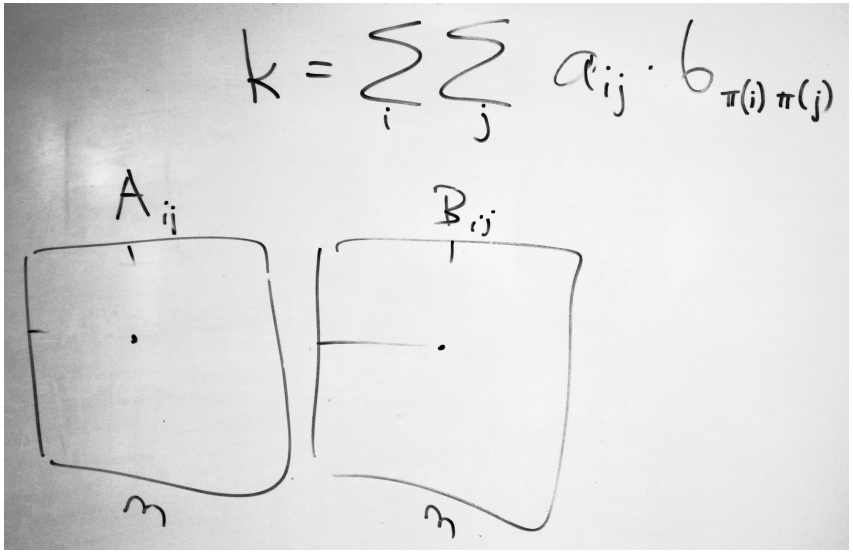
Which neighborhood will make it easier to optimize TSP?

The fitness landscape should be smooth... FDC should be high...

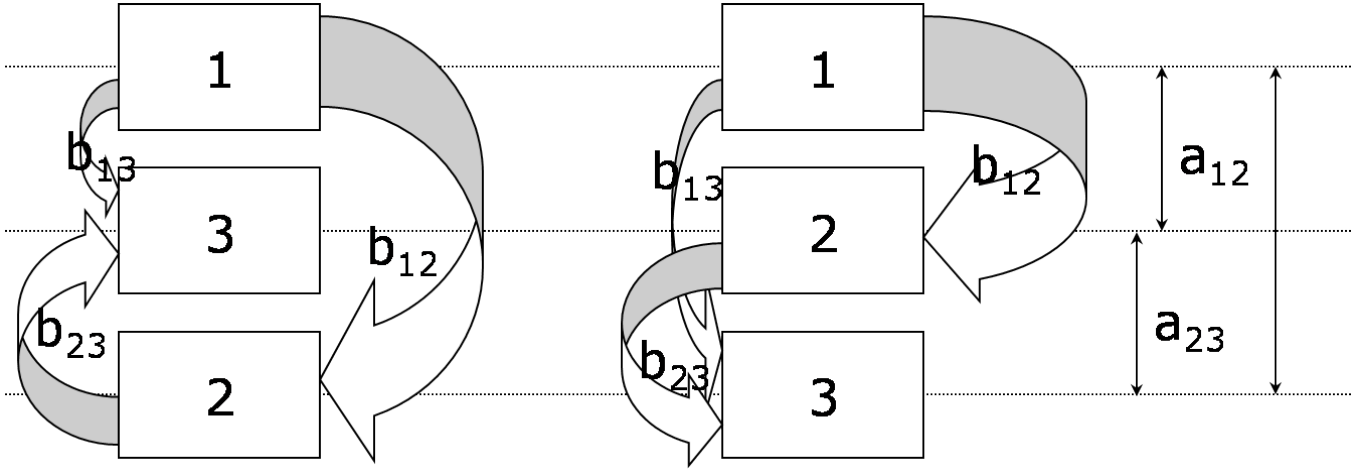


	STSP	ATSP
swapping cities		
path reversal		

Neighborhood in QAP (*quadratic assignment problem*)

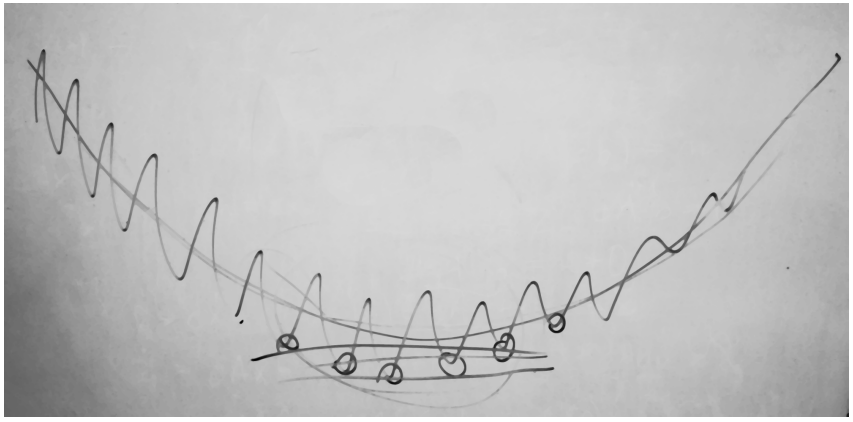


Modifying the assignment, for example 1-3-2 $\rightarrow$ 1-2-3:



Which neighborhood will make it easier to optimize TSP?

The fitness landscape should be smooth... FDC should be high...



	STSP	ATSP	QAP
swapping cities			
path reversal			

Neighborhood in GPP

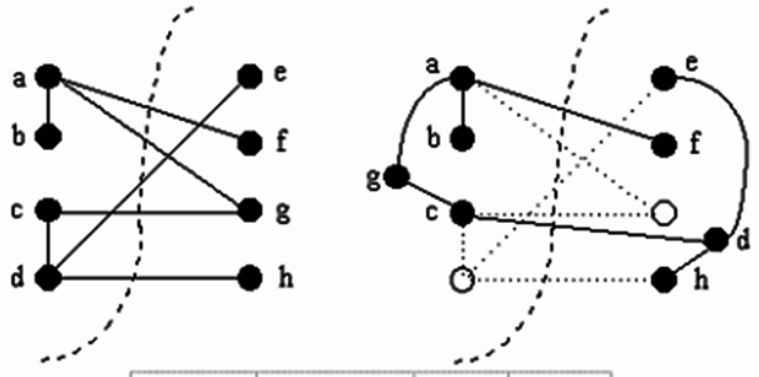
Neighborhood – all partitions (V_1', V_2') such that

$V_1' = V_1 \cup \{x\}$ and $V_2' = V_2 \backslash \{x\}$

or

$V_1' = V_1 \backslash \{y\}$ and $V_2' = V_2 \cup \{y\}$

$x \in V_2, y \in V_1$



Step No.	Vertex Pair	Gain	Cutset
0		0	5
1	(d, g)	3	2
2	(c, f)	1	1
3	(b, h)	-2	3
4	(a, e)	-2	5

Alternative approach:
generating infeasible solutions, i.e., $|V_1| \neq |V_2|$.

$$F(V_1, V_2) = \sum_{i \in V_1, j \in V_2} E_{ij} + \gamma(|V_1| - |V_2|)^2$$

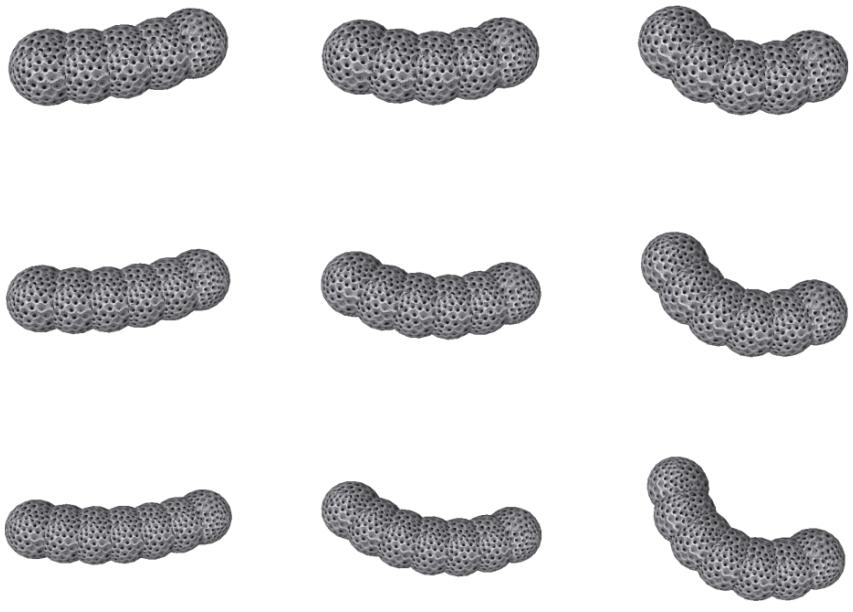
γ – a positive constant (a weight of a penalty for an infeasible partition).

Neighborhood of a vector of numbers

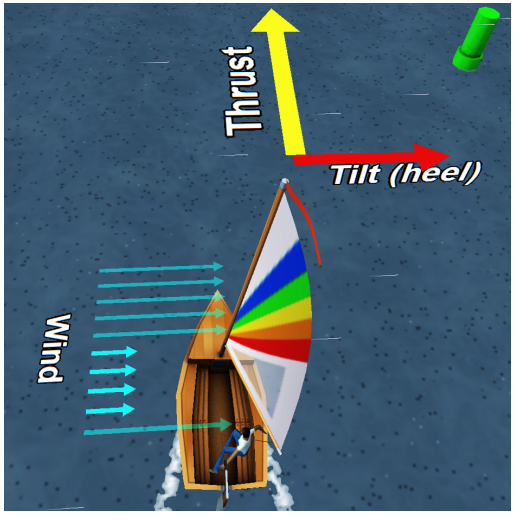
$$[4, 5.017, 3.422, -12.430, 107.819, ...]$$

Neighborhood of a vector of numbers

$$[4, 5.017, 3.422, -12.430, 107.819, ...]$$



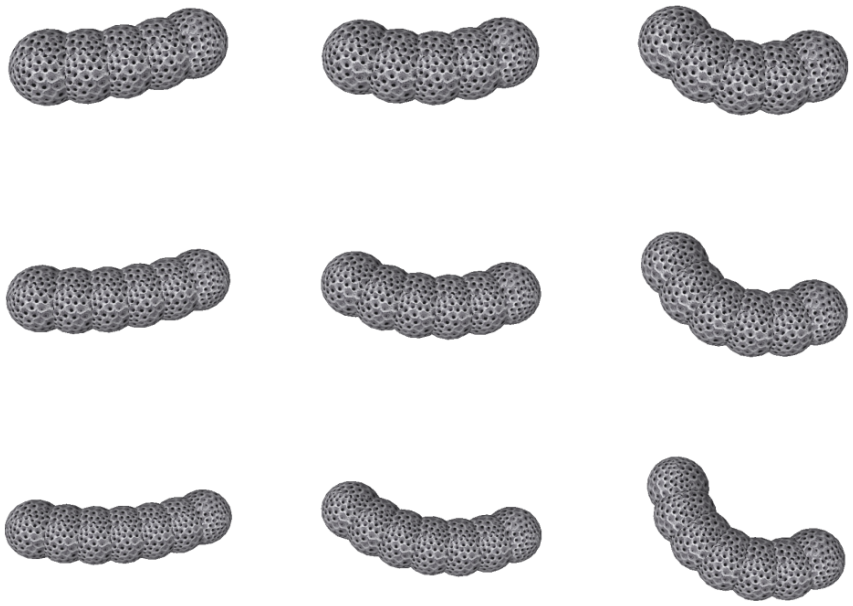
http://www.framsticks.com/foraminifera



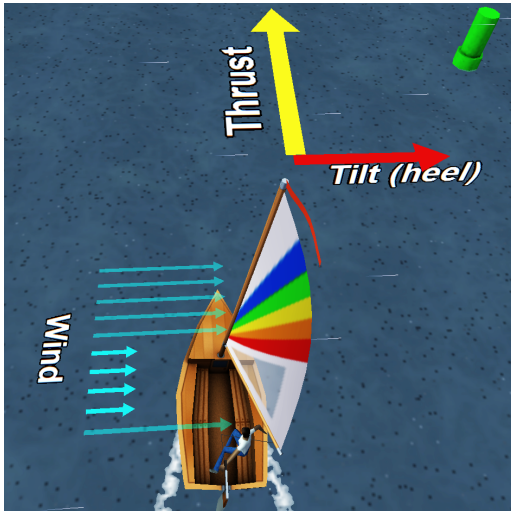
http://sailor.mooncoder.com/

Neighborhood of a vector of numbers

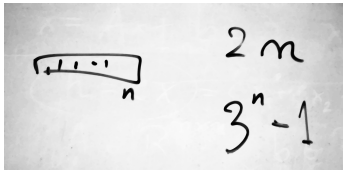
$$[4, 5.017, 3.422, -12.430, 107.819, ...]$$



http://www.framsticks.com/foraminifera



http://sailor.mooncoder.com/



Exploration of the neighborhood of the solution x

1. select a solution in S, evaluate it, define it as *current*
2. generate new solutions as neighbors of the current one and evaluate them
3. if a new solution is better, consider it the current one, otherwise ignore it
4. repeat steps 2 and 3 as long as there is improvement

Local search procedure

procedure LOCAL_SEARCH;
begin
    INITIALIZE(x_{start});
    $x_{current} := x_{start}$;
    repeat
        GENERATE(y from N($x_{current}$));
        if $f(y) \leq f(x_{current})$ **then**
            $x_{current} := y$;
    until $f(y) > f(x_{current})$ for all $y \in$N($x_{current}$);
end;

Local search procedure

procedure LOCAL_SEARCH;
begin
    INITIALIZE(x_{start});
    $x_{current} := x_{start}$;
    repeat
        GENERATE(y from N($x_{current}$)); $\longleftarrow$ implementation?
        if $f(y) \leq f(x_{current})$ **then** $\longleftarrow \leq$?
            $x_{current} := y$;
    until $f(y) > f(x_{current})$ for all $y \in$N($x_{current}$); $\longleftarrow >$?
end;

Local optimum

x_{min} is a *local minimum* if

$$\forall_{y \in N(x_{min})} \quad f(x_{min}) \leq f(y)$$

x_{max} is a *local maximum* if

$$\forall_{y \in N(x_{max})} \quad f(x_{max}) \geq f(y)$$

Local optimum

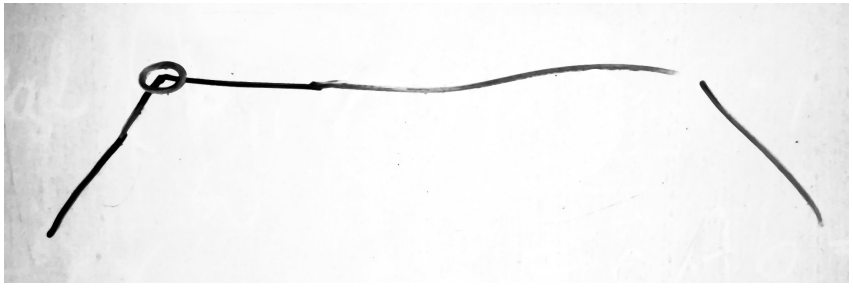
x_{min} is a *local minimum* if

$$\forall_{y \in N(x_{min})} \quad f(x_{min}) \leq f(y)$$

x_{max} is a *local maximum* if

$$\forall_{y \in N(x_{max})} \quad f(x_{max}) \geq f(y)$$

strict vs. non-strict inequality!



Two variants of Local Search

- Greedy – first improvement; first descent
- Steepest – best improvement; highest descent

Two variants of Local Search

- Greedy – first improvement; first descent
- Steepest – best improvement; highest descent

Could you implement LS now?

Example (maximization in a 2D grid)

http://en.alife.pl/opt/e/index.html

x1 \ x2	**1**	**2**	**3**	**4**	**5**
A	1	2	1	7	4
B	0	2	1	2	4
C	1	3	3	4	8
D	3	4	5	3	3
E	6	5	3	2	1

- start from B2
- simulate Greedy and Steepest
- consider two neighborhoods: Moore (8) and von Neumann (4)
- think about the influence of the condition $<$ vs. $\leq$
- formulate conclusions

Efficient calculation of the cost of a new solution

- TSP time: linear (naive) or constant (easy)
- QAP time: quadratic (naive) or linear (easy) or quasi-constant (auxiliary table)

Disadvantages and advantages

Disadvantages

- LS algorithms stop working (prematurely?) in the local optimum
- the quality of the solutions obtained may depend on the choice of the starting solution
  - for most problems, there are no guidelines on how the starting solution should be selected for best results

Disadvantages and advantages

Disadvantages

- LS algorithms stop working (prematurely?) in the local optimum
- the quality of the solutions obtained may depend on the choice of the starting solution
  - for most problems, there are no guidelines on how the starting solution should be selected for best results

Advantages

- flexible
- can be used for any combinatorial optimization problem
- very fast and simple

Avoiding the disadvantages of local search algorithms

- running the local search algorithm for a large number of different starting solutions
  - multi-random start variant
  - guided local search variant: the goal function is modified in subsequent runs so that parts of the solutions existing in previously found local optima are penalized
- introduction of a more complex definition of neighborhood in order to search a larger part of the space of feasible solutions
- changing the neighborhood definition used while running – *variable neighborhood search*
- to some extent, accepting the deterioration of the value of the goal function
- ...