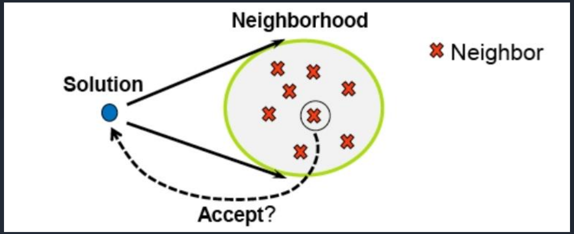
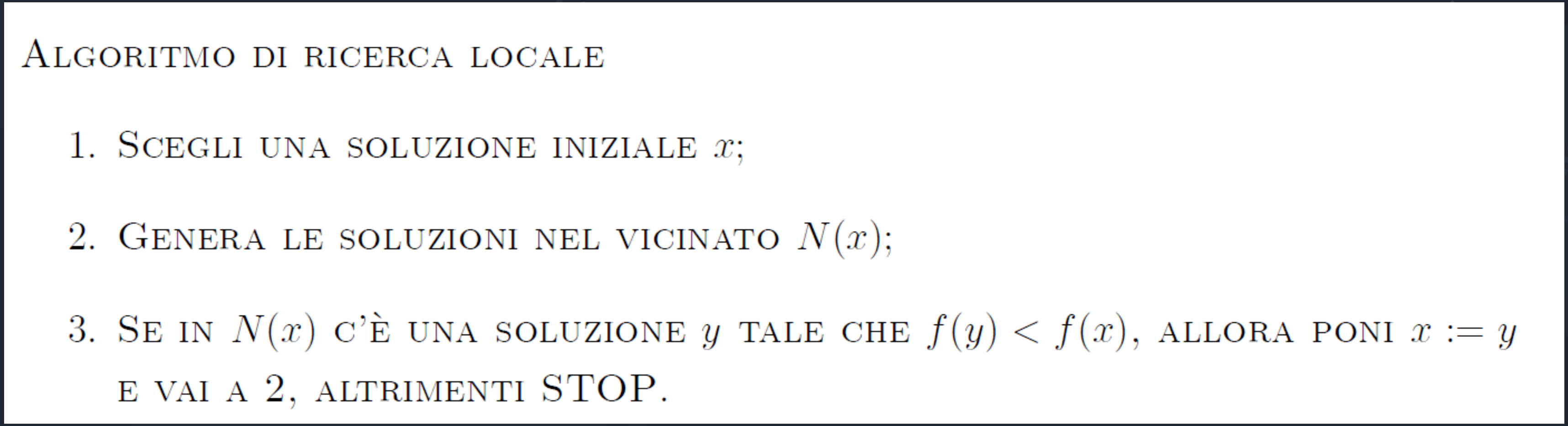
Local Search

Local Search

- Consider a minimization problem, and its admissible solution x, with associated value of the objective function f(x).
- Local search consists of defining a neighborhood of x (called, in local search terminology, neighborhood), and exploring it in search of better solutions, if any.
- If, in this neighborhood of x, we discover a solution y for which f(y) < f(x), then we move from x to y and we start again from y with the exploration of its neighborhood. If instead in the neighborhood of x no better solution is discovered, then it means that x is a local minimum.
- In classical local search, arrived in a local minimum, the algorithm stops and returns this minimum as an output value.





```
ALGORITMO DI RICERCA LOCALE

1. SCEGLI UNA SOLUZIONE INIZIALE $x$;
2. GENERA LE SOLUZIONI NEL VICINATO $N(x)$;
3. SE IN $N(x)$ C'È UNA SOLUZIONE $y$ TALE CHE $f(y) < f(x)$, ALLORA PONI $x := y$
   E VAI A 2, ALTRIMENTI STOP.
```

TWO BASIC DEFINITIONS

- A **neighborhood** C(s) around a solution s is a function that returns a subset of solutions that are somehow close or related to s
  - implies a "direction of search"
  - multiple neighborhoods can be defined for a solution s
  - could be exhaustive or not (if not, it could actually be random)

- A **move** $s \longleftarrow s'$ constitutes a choice of a solution $s' \in C(s)$

First improvement vs steepest descent

Come viene esplorato il vicinato di una soluzione? Due strategie

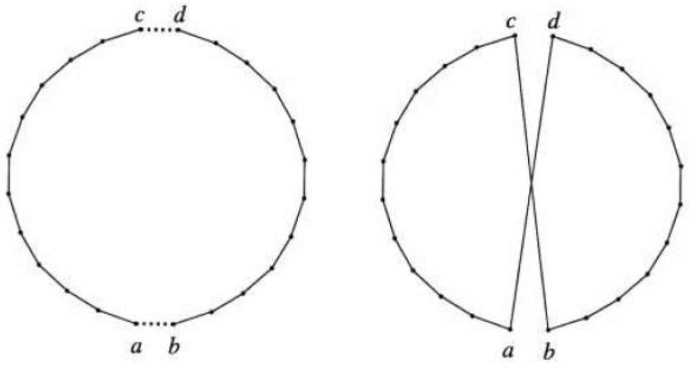
- First improvement: l'esplorazione del vicinato termina non appena si trova una soluzione migliore di quella corrente.
- Steepest descent: esplora comunque tutto cercando il massimo miglioramento che quel vicinato consente di ottenere.

In genere si preferisce il primo approccio, ma non è una regola fissa.

TSP: local search

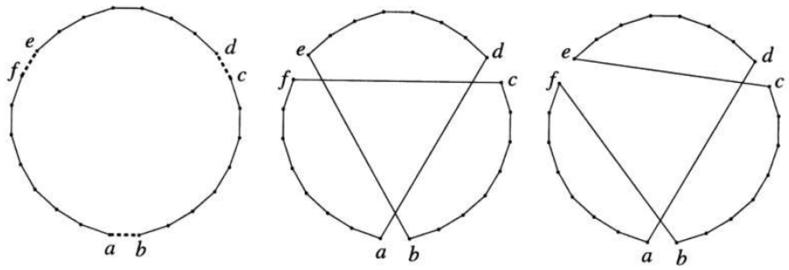
1. Start from complete solution (a tour)
2. Compute the best (the first) edge exchange that improves the tour.
3. Execute this exchange
4. Search for another exchange until no improvement is possible

2-opt



GAIN = (a,d)+(b,c)-(c,d)-(a,b)
Subpath (b,...,d) is reverted

3-opt



Two possible new tours
GAIN1 = (a,d)+(e,b)+(c,f)-(a,b)-(c,d)-(e,f) no path is reverted
GAIN2 = (a,d)+(e,c)+(b,f)-(a,b)-(c,d)-(e-f) path (c,...,b) is reverted

2-opt 3opt : results for random problems

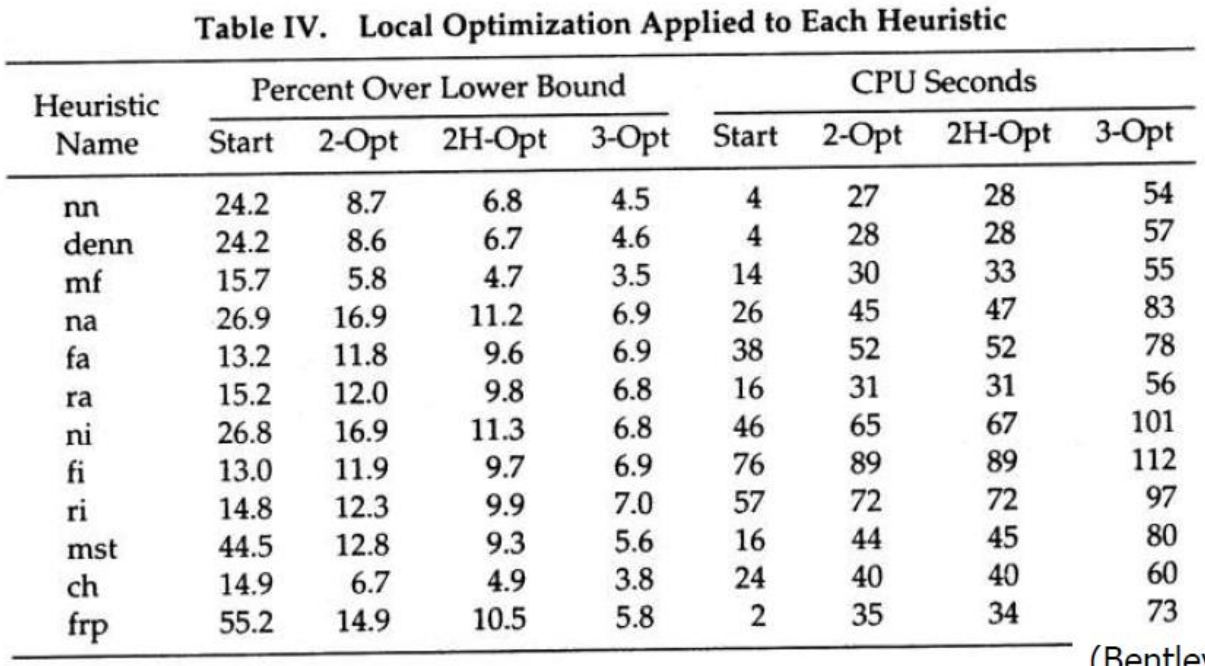


Table IV. Local Optimization Applied to Each Heuristic

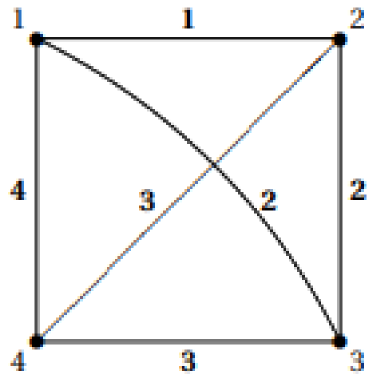
Heuristic Name	Percent Over Lower Bound				CPU Seconds			
	Start	2-Opt	2H-Opt	3-Opt	Start	2-Opt	2H-Opt	3-Opt
nn	24.2	8.7	6.8	4.5	4	27	28	54
denn	24.2	8.6	6.7	4.6	4	28	28	57
mf	15.7	5.8	4.7	3.5	14	30	33	55
na	26.9	16.9	11.2	6.9	26	45	47	83
fa	13.2	11.8	9.6	6.9	38	52	52	78
ra	15.2	12.0	9.8	6.8	16	31	31	56
ni	26.8	16.9	11.3	6.8	46	65	67	101
fi	13.0	11.9	9.7	6.9	76	89	89	112
ri	14.8	12.3	9.9	7.0	57	72	72	97
mst	44.5	12.8	9.3	5.6	16	44	45	80
ch	14.9	6.7	4.9	3.8	24	40	40	60
frp	55.2	14.9	10.5	5.8	2	35	34	73

(Bentley 1992)

NN=Nearest Neighbourhood, *DENN*=Double Ended NN, *MF*=Multiple Fragment, *NA, FA, RA*=Nearest, Farthest, Random Addition, *NI, FI, RI*=Nearest, Farthest, Random Insertion, *MST*=Min. Spanning Three, *CH*=Christofides, *FRP*=Fast Recursive Partition.

PROBLEMA DELLA PARTIZIONE UNIFORME SU GRAFI

- Si consideri un grafo G = (V;E) non orientato, con 2n nodi, pesato sugli archi. Una partizione (A;B) dell'insieme dei nodi si dice uniforme se |A| = |B|.
- Il costo di una partizione uniforme e' il peso complessivo degli archi che non sono interamente contenuti in una classe della partizione, ovvero degli archi che cadono a cavallo dei due insiemi.
  Minimizzare il costo.



Graph Partition: local search 2 swap

Given a uniform partition (A;B), consider a node a from A and a node b from B. Swap them.

The neighborhood of a partition (A;B) as the set of partitions that can be obtained by swapping in all ways.

The size of this neighborhood is O(n^2).

This choice yielded quite satisfactory results.

Graph Partition: local search 4 swap

Given a uniform partition (A;B), consider 2 nodes from A and 2 nodes from B. Swap them.

The neighborhood of a partition (A;B) as the set of partitions that can be obtained by swapping in all ways.

The size of this neighborhood is O(n^4).

This choice yielded better results but it is too expensive.

Graph Partition: local search Lin Kernighan

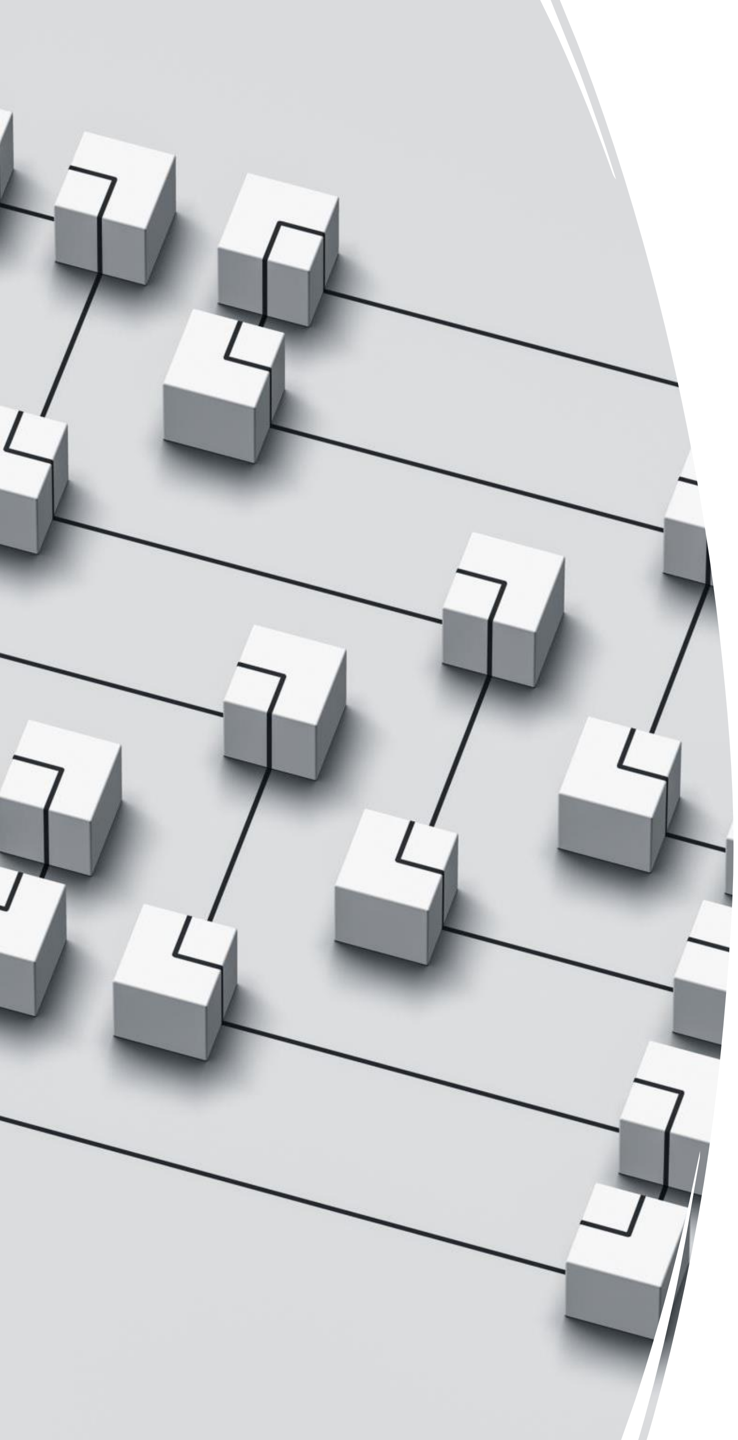
Given a uniform partition (A;B), consider the best single swap (a,b), make it. This gives a new partition (A',B'). From now on don't swap (a,b) anymore.

Repeat the above step starting from the last generated partition n times. Return the best partition.

The size of this neighborhood is O(n^3).

The algorithm stops when, among all the n partitions individuated in the intermediate steps, none results strictly better than the current one.

Great experimental results!!

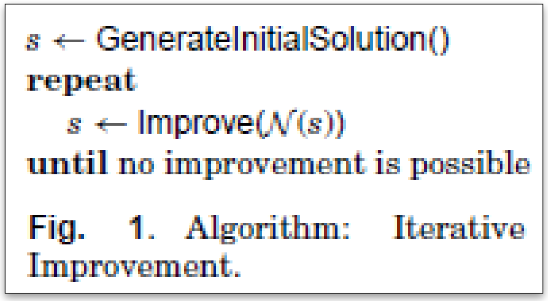


Exercises

Design a local search algorithm for the following problems:

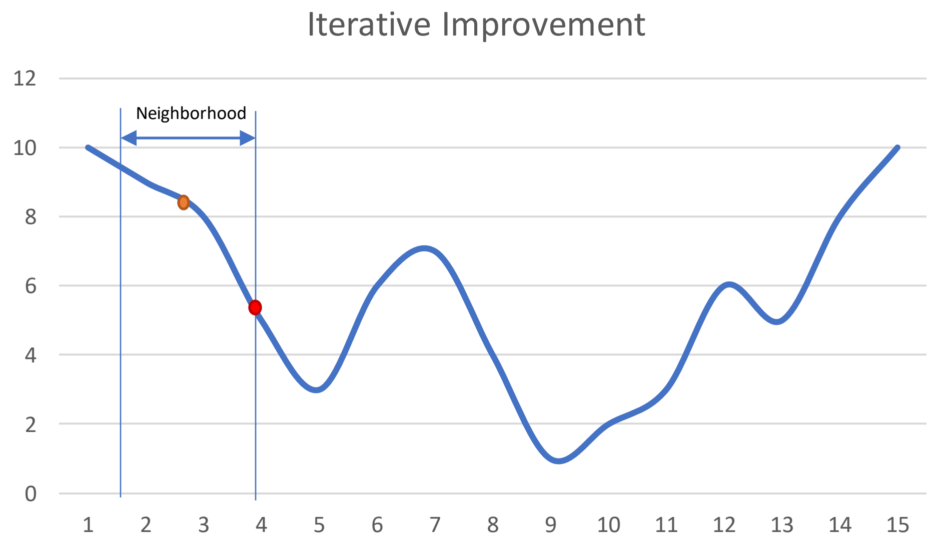
- Independent set
- Graph coloring
- Scheduling identical parallel machines

BASIC LOCAL SEARCH



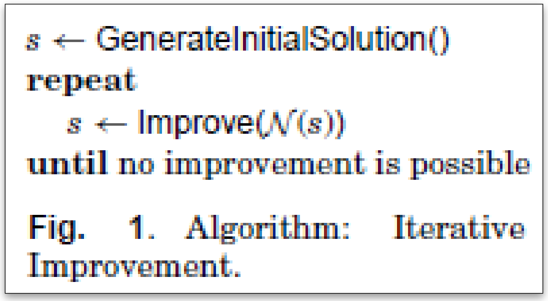
```
$s \leftarrow$ GenerateInitialSolution()
repeat
    $s \leftarrow$ Improve($\mathcal{N}(s)$)
until no improvement is possible
```

Fig. 1. Algorithm: Iterative Improvement.



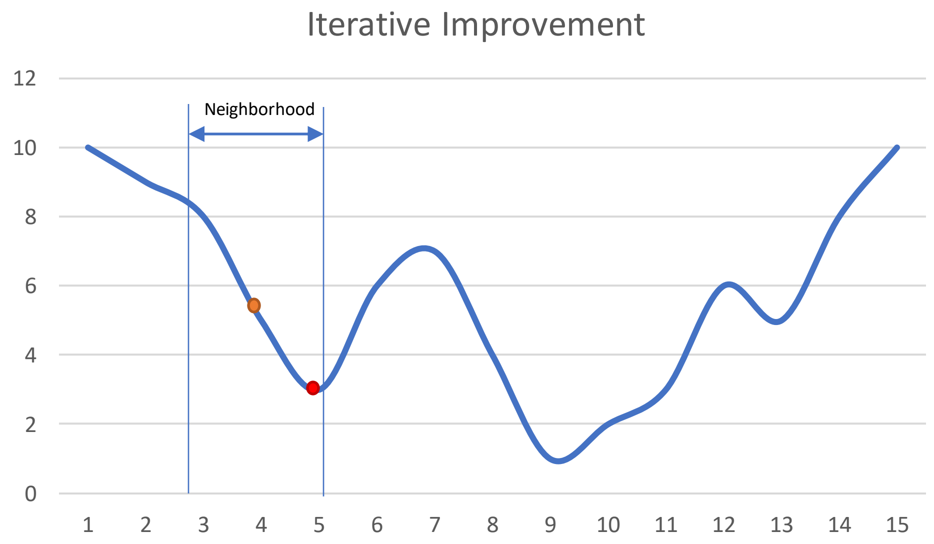
Improve() can be implemented as “first improvement” or “best improvement” or anything in between

BASIC LOCAL SEARCH

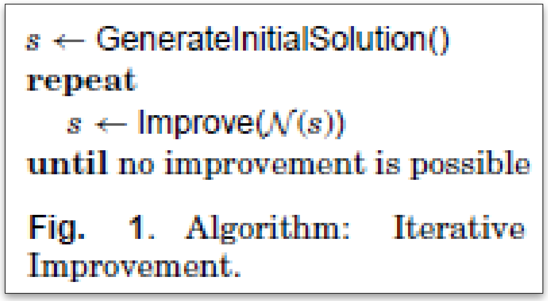


```
$s \leftarrow$ GenerateInitialSolution()
repeat
    $s \leftarrow$ Improve($\mathcal{N}(s)$)
until no improvement is possible
```

Fig. 1. Algorithm: Iterative Improvement.

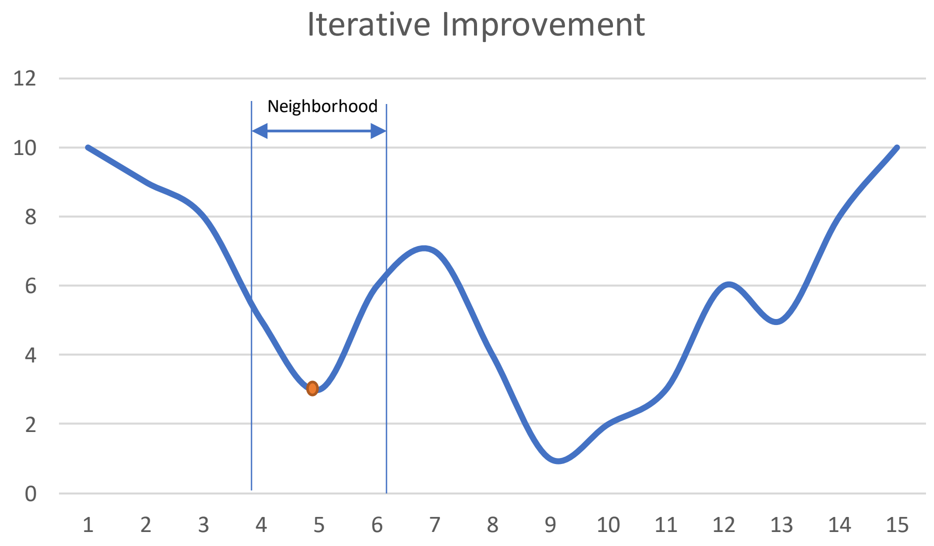


BASIC LOCAL SEARCH



```
$s \leftarrow$ GenerateInitialSolution()
repeat
    $s \leftarrow$ Improve($\mathcal{N}(s)$)
until no improvement is possible
```

Fig. 1. Algorithm: Iterative Improvement.



Local optimum found, cannot improve!

How to escape the local optimum "trap"?

From Local Search to Metaheuristics

Local search is not able to escape from local minimum

Sometimes the neighborhood is too large

A way to solve it is the following

- Stochastically explore only a subset of the neighborhood
- Accept solutions that are worst than the previous