OPTIMIZATION TECHNIQUES

Variable neighborhood search

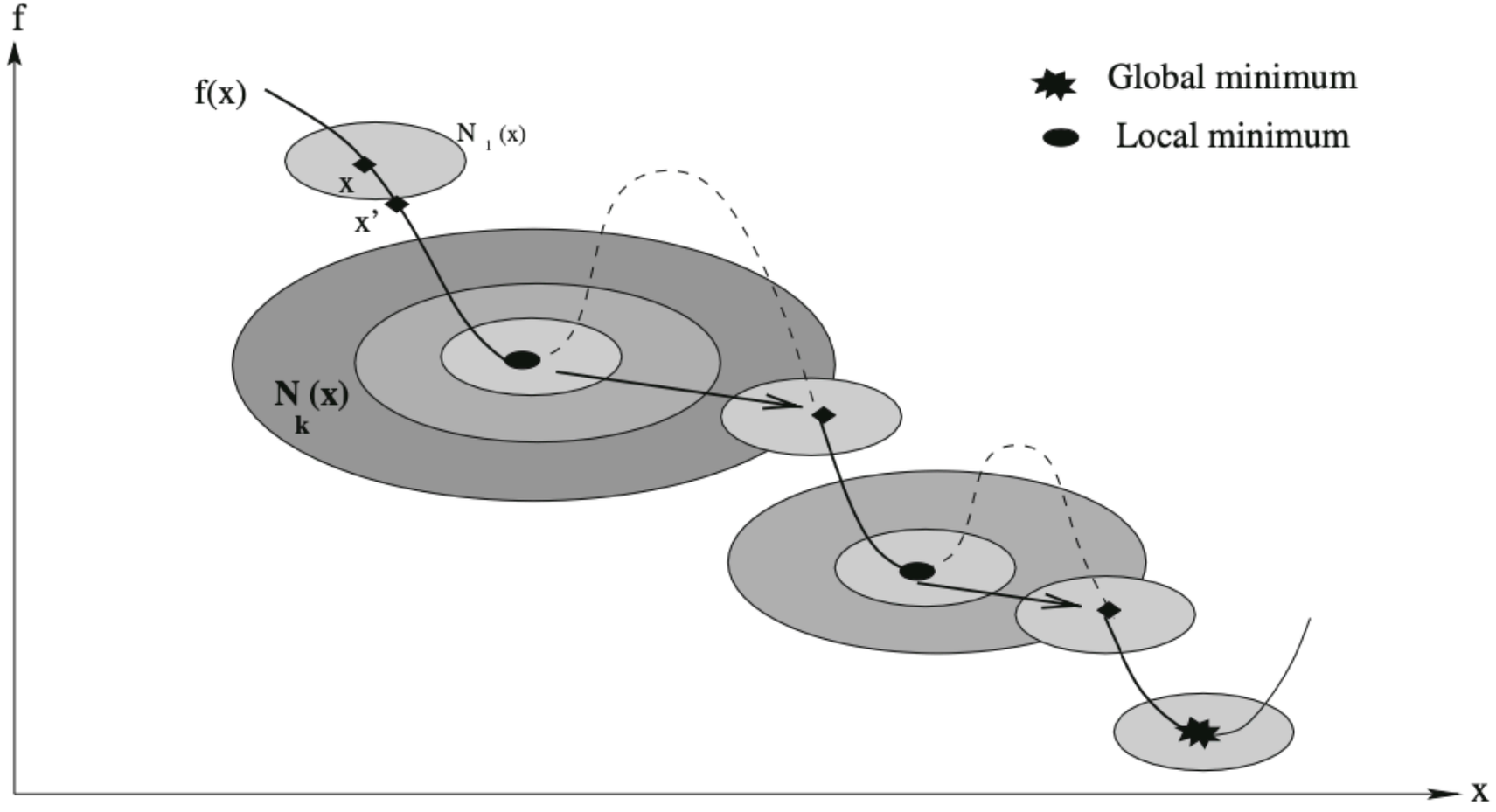
Prof. Giovanni Iacca

giovanni.iacca@unitn.it

UNIVERSITY OF TRENTO - Italy

Information Engineering
and Computer Science Department

Variable Neighborhood Search



VARIABLE NEIGHBORHOOD SEARCH

LET US FIRST CONSIDER TWO GENERAL SCHEMES FOR LOCAL SEARCH

Function `BestImprovement(`x`)`

1 **repeat**
2 $\quad x' \leftarrow x$
3 $\quad x \leftarrow arg\min_{y \in N(x)} f(y)$
until $(f(x) \geq f(x'))$
return x

Function `FirstImprovement(`x`)`

1 **repeat**
2 $\quad x' \leftarrow x; \quad i \leftarrow 0$
3 $\quad$ **repeat**
4 $\quad\quad i \leftarrow i+1$
5 $\quad\quad x \leftarrow arg\min\{f(x), f(x^i)\}, \quad x^i \in N(x)$
$\quad$ **until** $(f(x) < f(x')$ `or` $i = |N(x)|)$
until $(f(x) \geq f(x'))$
return x

- $N(x)$ is the set of solutions in a given neighborhood of the incumbent solution x

- In combinatorial optimization (discrete variables), it is possible to enumerate all the solutions in a given neighborhood of x (e.g., in binary problems, all solutions that differ for 1 bit). In continuous optimization, one may e.g. define a minimum perturbation ε along each direction and perturb each variable in x as $\pm\ \varepsilon$. Many other neighborhood structures can be defined!
  - **BestImprovement** (a.k.a. steepest descent): visits all solutions in the neighborhood
    → exhaustive search, expensive!
  - **FirstImprovement** (a.k.a. first descent): accepts first improved solution, continue from there

VARIABLE NEIGHBORHOOD SEARCH

MAIN GIST OF THE ALGORITHM

- Originally proposed by Mladenović & Hansen for combinatorial problems:
  Nenad Mladenović; Pierre Hansen (1997). "Variable neighborhood search". Computers and Operations Research. 24 (11): 1097–1100.

- Later on extended also for continuous problems, active research

- Systematic change of the neighborhood during the search

- Does not follow a single trajectory but explores increasingly distant neighbors of the incumbent solution (i.e., the starting point of the local search)

- Jumps from this solution to a new one if and only if there is an improvement

- In this way, keeps good (maybe optimal) variables in the incumbent and obtains promising neighbors

- Uses local search to get from these neighbors to local optima

… *remember Basin Hopping?*

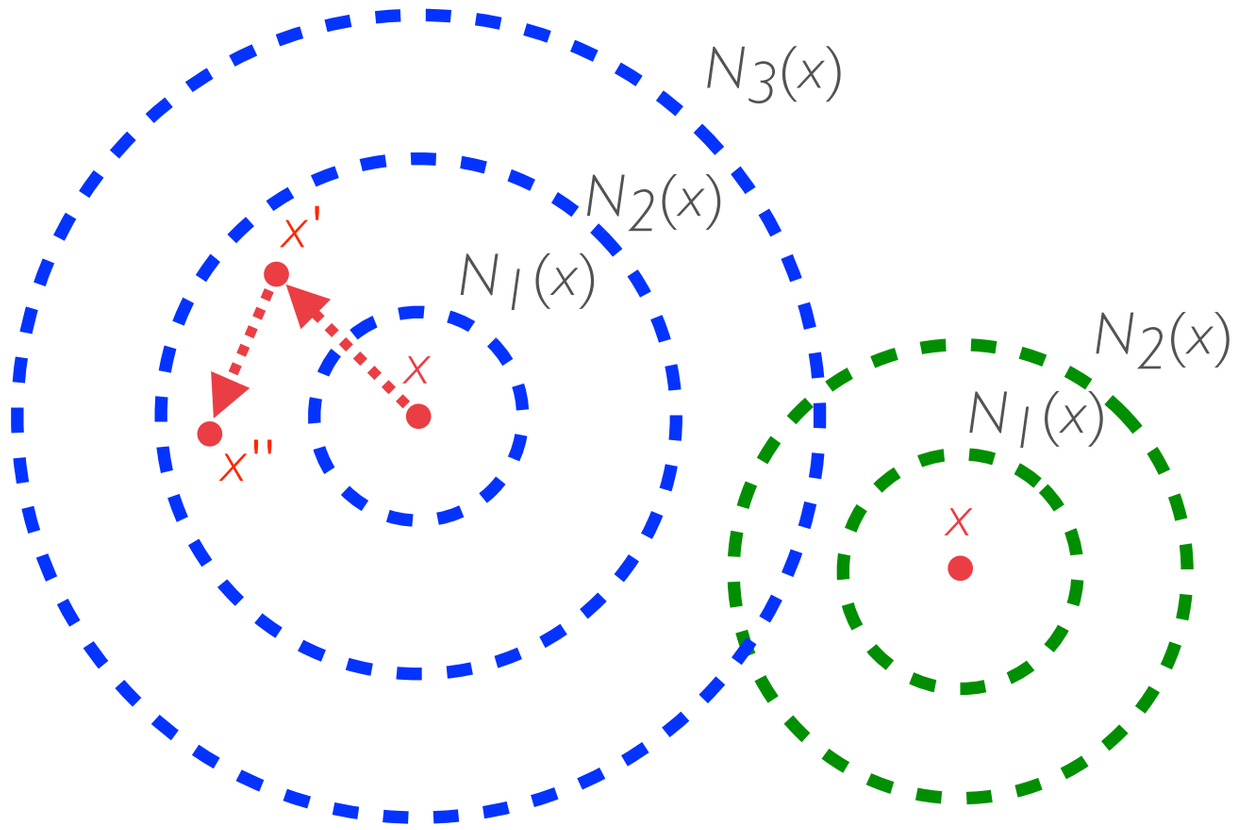
VARIABLE NEIGHBORHOOD SEARCH

VNS: INTUITION

- A local minimum with respect to one neighborhood structure is not necessary so for another
  - Try to escape from local optima trap by changing the neighborhood structure
  - Many neighborhood structures are associated with distance between solutions

- Only the global minimum is a local minimum for all possible neighborhood structures

- For many problems local minima with respect to one or several neighborhoods are relatively close to each other



VARIABLE NEIGHBORHOOD SEARCH

VNS: BASIC ALGORITHM

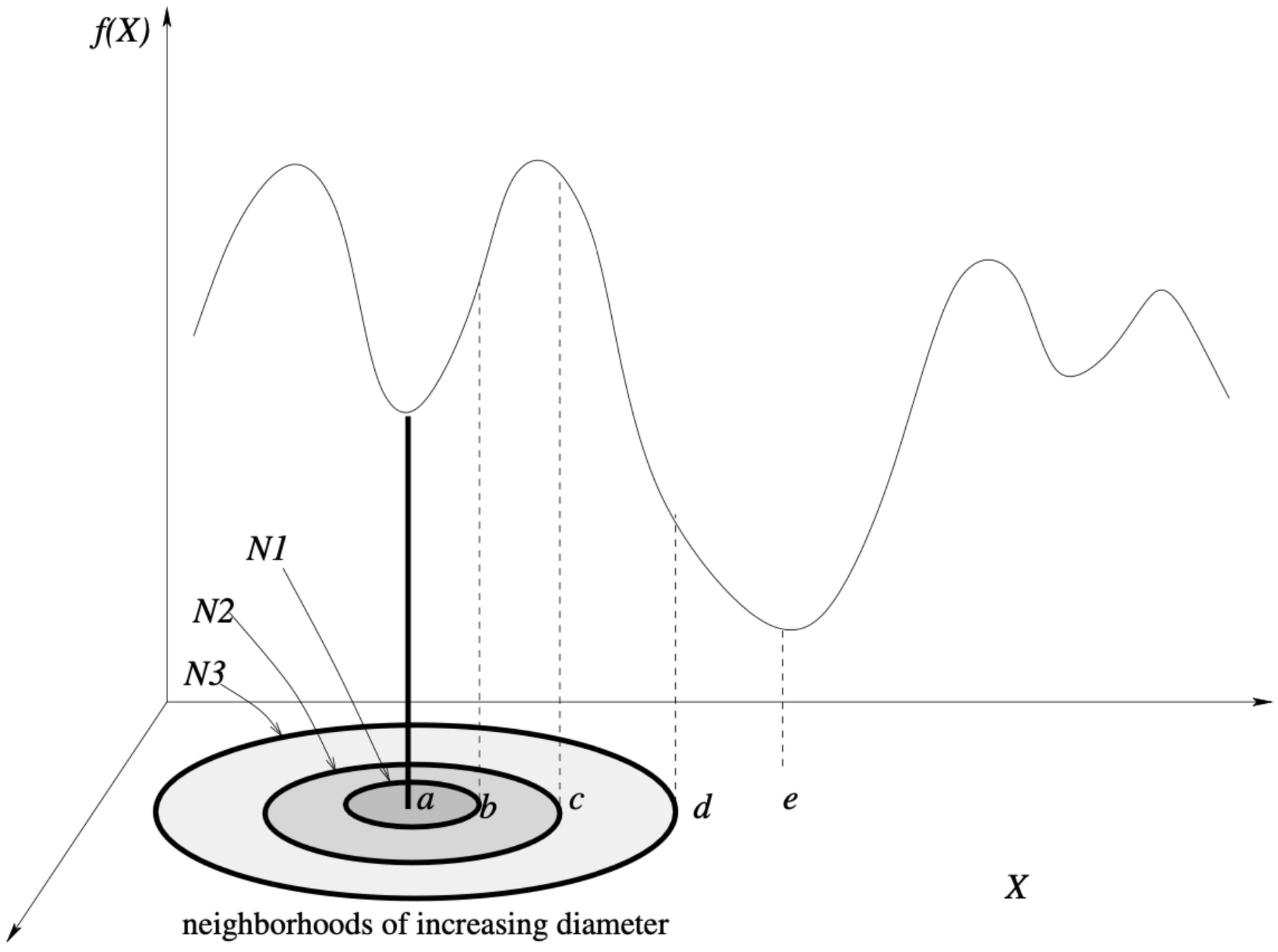
- Initialization: Select a set of neighborhood structures N_k ($k=1,\ldots,k_{max}$), find an initial solution x, set $k=1$, choose a stopping condition
- Step 1 (**Shaking**): Generate a random $x' \in N_k(x)$ (to avoid cycling)
- Step 2 (**Local Search**): Apply a local search method starting with x' to find local optimum x''
  - Different implementations of VNS use as LS either **FirstImprovement** or **BestImprovement**
- Step 3 (**Move or Not**): If x'' is better than the incumbent x, then set $x=x''$ and $k=1$, otherwise set $k=k+1$ (or if $k=k_{max}$ set $k=1$)
- Go back to Step 1

NOTES

- Successive N_k are often nested
  - Why? In general, neighborhoods should be progressively increasing in size, in order to allow a controllable amount of "intensification" (using the first neighborhoods) or "diversification" (using the last ones) → the usual "exploration vs exploitation tradeoff", but with more emphasis on LS: the principle is "intensification first, minimal diversification only if needed"
- In Step 3, if incumbent x is changed, then start over with N_1, otherwise continue search in N_{k+1} starting with the local optimum of N_k

VARIABLE NEIGHBORHOOD SEARCH

VNS: BASIC ALGORITHM



VARIABLE NEIGHBORHOOD SEARCH

VNS: BASIC ALGORITHM

Procedure Basic VNS
    select the neighborhood set
    find an initial solution
    choose a stopping condition
    repeat until stopping condition is met:
        set the first neighborhood
        repeat until the last neighborhood
            Shaking()
            LocalSearch()
            MoveOrNot()
End

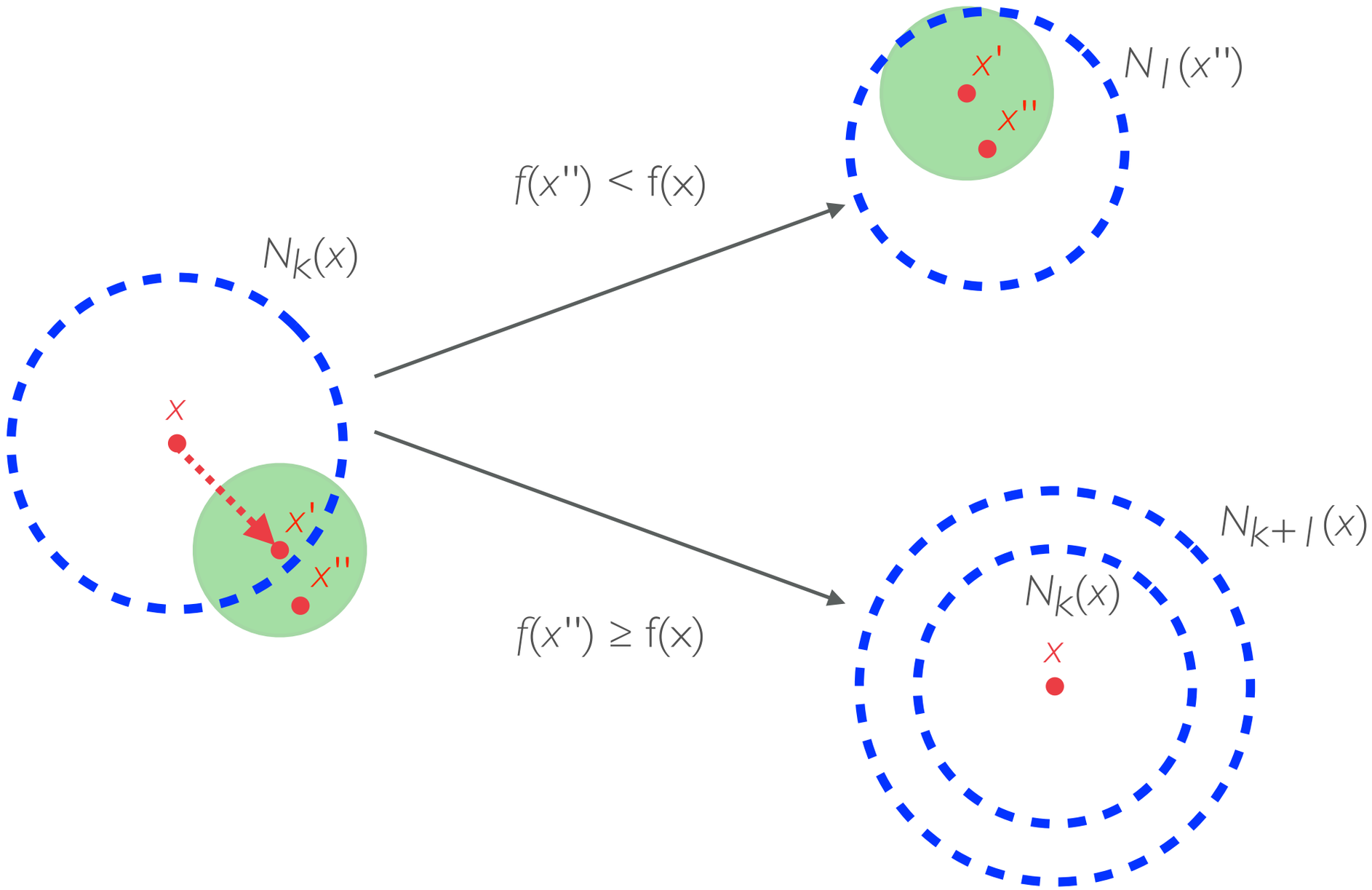
VARIABLE NEIGHBORHOOD SEARCH

VNS: BASIC ALGORITHM

Procedure Basic VNS
    select $\{N_k\}$, k = 1, ..., k_{max}
    find an initial solution x
    choose a stopping condition
    repeat until stopping condition is met:
        k ← 1
        repeat until k = k_{max}
            x' ← RandomSolution(N_k (x))
            x'' ← LocalSearch(x')
            if f(x'') < f(x) then
                x ← x''
                k ← 1
            else
                k ← k + 1
End

VARIABLE NEIGHBORHOOD SEARCH

VNS: BASIC ALGORITHM



VARIABLE NEIGHBORHOOD SEARCH

EXAMPLE - BINARY PROBLEMS

- For example, if the search space is given by binary strings, one may consider as N1 all changes of a single bit, N2 all changes of two bits, etc.

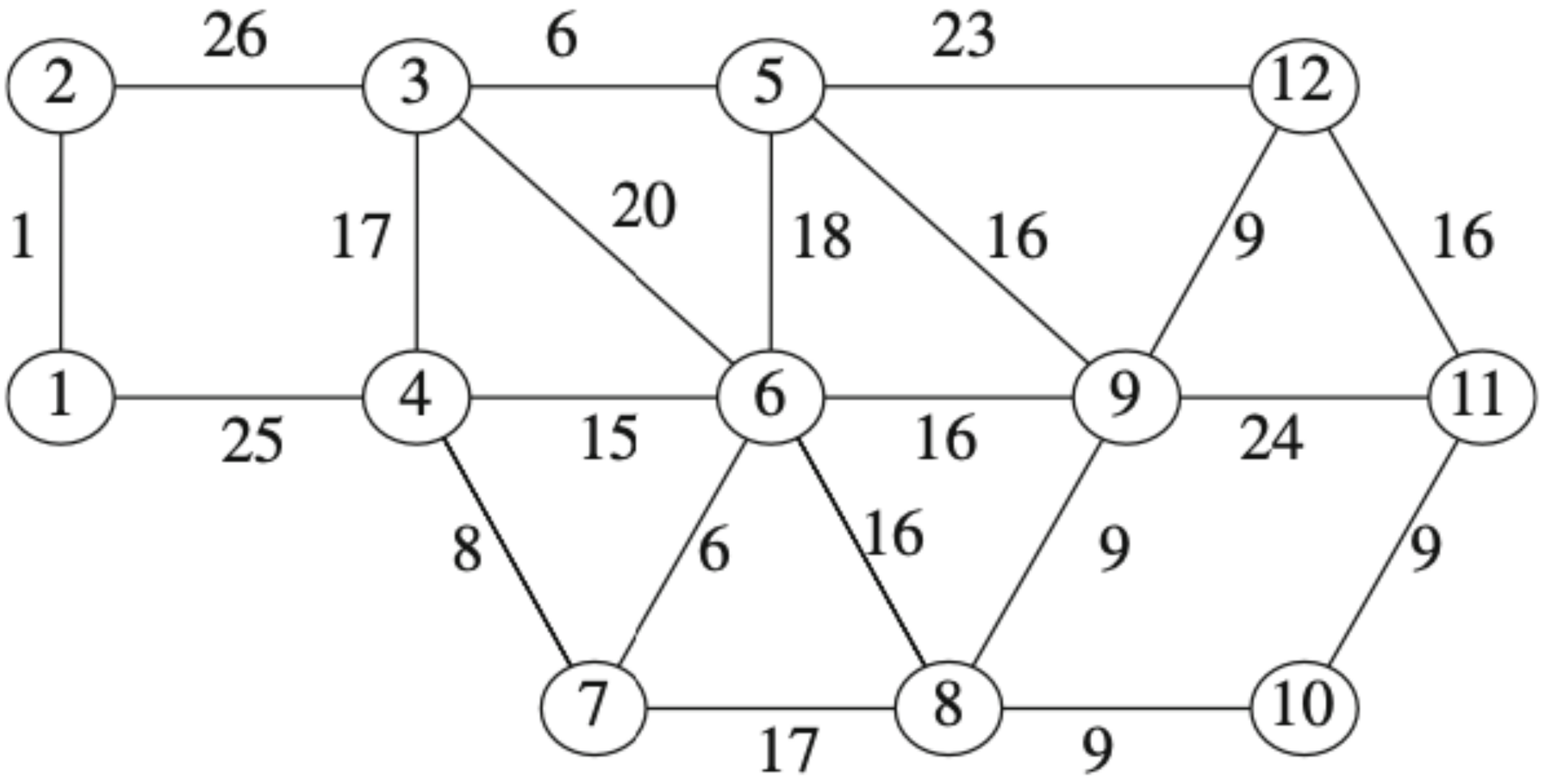
- If local search makes progress one sticks to the default neighborhood N1.

- As soon as a local minimum with respect to N1 is encountered, one tries to go to greater Hamming distances from the current point, aiming at discovering a nearby attraction basin, possibly leading to a better local optimum.

VARIABLE NEIGHBORHOOD SEARCH

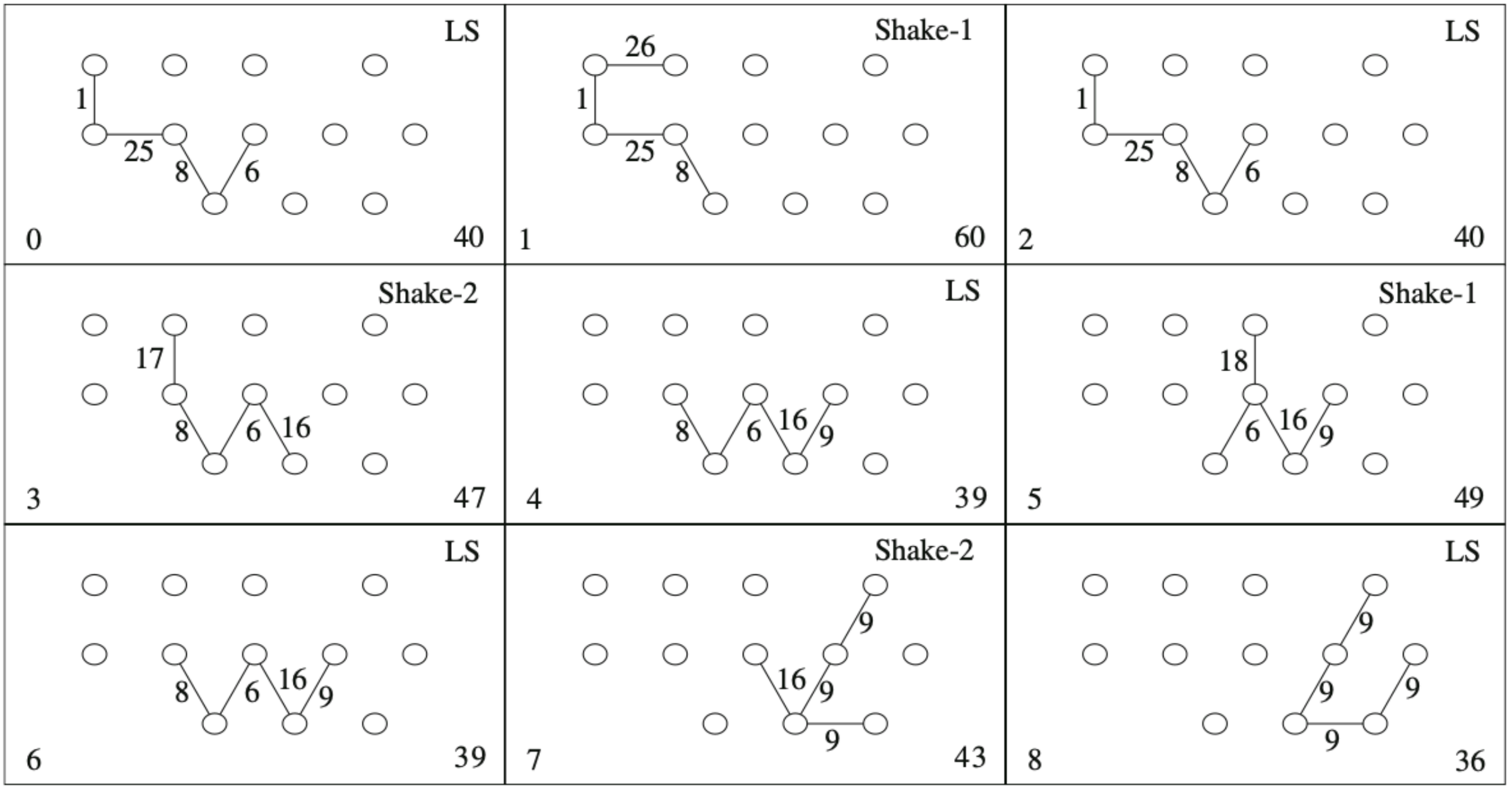
EXAMPLE - K-CARDINALITY TREE

Goal: The minimum k-cardinality tree problem on graph G (k-card for short) consists in finding a subtree of G with exactly k edges whose sum of weights is minimum.



VARIABLE NEIGHBORHOOD SEARCH

EXAMPLE - K-CARDINALITY TREE (K=4)



VARIABLE NEIGHBORHOOD SEARCH

VNS: ALGORITHMIC CHOICES

How to generate the various neighborhood structures?

- For many problems different neighborhood structures (local searches) exist
- Change parameters of existing local search algorithms

Several questions about selection of neighborhood structures are in order:

- What properties of the neighborhoods are mandatory for the resulting scheme to be able to find a globally optimal or near-optimal solution?
- What properties of the neighborhoods will favor finding a near-optimal solution?
- Should neighborhoods be nested? Otherwise how should they be ordered?
- What are desirable properties of the sizes of neighborhoods?

VARIABLE NEIGHBORHOOD SEARCH

VNS: ALGORITHMIC CHOICES

- Number and types of neighborhoods to be used

- Order of their use in the search
  - Forward VNS: start with $k=1$ and set $k=k+1$ if no better solution is found, otherwise set $k=1$
  - Backward VNS: start with $k=k_{max}$ and set $k=k-1$ if no better solution is found

- Strategy for changing the neighborhoods

- Local search method(s)

- Stopping condition

NOTES

- Usually, no need to design sophisticated acceptance criteria to escape from local optima
- A good design of neighborhoods is crucial for VNS

VARIABLE NEIGHBORHOOD SEARCH

VARIANT: VARIABLE NEIGHBORHOOD DESCENT (VND)

- Initialization: same as before
- Step 1: find the best $x' \in N_k(x)$ (**BestImprovement** scheme)
- Step 2: if x' is better than x, then set $x=x'$, otherwise set $k=k+1$

NOTES

- Implies a deterministic change of the neighborhood during the search
- Meaningful as a local optimum of one neighborhood is not necessarily one in another

```
Procedure VND
    select {N_k}, k = 1, ..., k_max
    find an initial solution x
    k ← 1
    repeat until k > k_max
        x' ← FindBestNeighbour(N_k (x))
        if f(x') < f(x) then
            x ← x'
            k ← 1
        else
            k ← k + 1
End
```

VARIABLE NEIGHBORHOOD SEARCH

VARIANT: REDUCED VNS (RVNS)

- Same as basic VNS except that no local search procedure is applied, to save time
- Step 1: random solutions are generated from increasingly larger neighborhoods of *x*
- Step 3: move iff the new solution is better

NOTE

- Can reach good quality solutions faster than basic VNS (even though no explicit descent is made!)
- Especially useful for very large instances for which local search is costly

```
Procedure Reduced VNS
    select {N_k}, k = 1, ..., k_max
    find an initial solution x
    choose a stopping condition
    k ← 1
    repeat until k = k_max
        x' ← RandomSolution(N_k (x))
        if f(x') < f(x) then
            x ← x'
            k ← 1
        else
            k ← k + 1
End
```

VARIABLE NEIGHBORHOOD SEARCH

VARIANT:VARIABLE NEIGHBORHOOD DECOMPOSITION SEARCH (VNDS)

- Step 1: all but *n* variables of solution *x* are fixed, choose these *n* variables at random
- Step 2: apply local search on an *n*-dimensional subproblem in the space of the unfixed variables
- Step 3: same as in the basic VNS
- Basic VNS can be used as the local search heuristic (two level recursive VNS)

```
Procedure VNDS
    select {N_k}, k = 1, ..., k_max
    find an initial solution x
    choose a stopping condition
    k ← 1
    repeat until k = k_max
        x' ← RandomSolution(N_k (x))
        t' ← DetermineFreeComponents(x')
        t'' ← LocalSearch(t')
        x'' ← InjectComponents(x',t'')
        if f(x'') < f(x) then
            x ← x''
            k ← 1
        else
            k ← k + 1
End
```

VARIABLE NEIGHBORHOOD SEARCH

VARIANT: SKEWED VNS (SVNS)

- Addresses the problem of exploring valleys far from the incumbent solution *x*:
  - Once the best solution in a large region is found, it may be necessary to go quite far to obtain an improved one
  - Solutions drawn at random in far-away neighborhoods may differ substantially from the incumbent and in those cases VNS can degenerate into a multi-start heuristic (in which descents are made iteratively from solutions generated at random, that is not very efficient)
- Steps 1 and 2 are the same as in the basic VNS
- Step 3: instead of the objective function, use an evaluation function that takes into account also the distance $\rho(x, x'')$ between x'' and x
  - Step 3 (improve or not): If $f(x'') < f(x_{best})$, set $x_{best}=x''$
  - Step 4 (move or not): If $f(x'')-\alpha\rho(x, x'') < f(x)$, set $x=x''$ and $k=1$, otherwise set $k=k+1$

NOTES

- α must be chosen in order to accept exploring valleys far from x when $f(x'')$ is larger than $f(x)$, but not too much (otherwise one will always leave x) → good value is problelm-dependent
- In order to avoid frequent moves from x to a close solution, one may take a large value for α when $\rho(x, x'')$ is small
- More sophisticated choices for $\rho(x,x'')$ can be made through some learning process

VARIABLE NEIGHBORHOOD SEARCH

OTHER POSSIBLE MODIFICATIONS

Changes in Step 1 (shaking)

- Generate a solution from each of the k_{max} neighborhoods and move to the best of them
- Choose the best of l randomly generated solutions from N_k

Changes in Step 3 (move or not)

- Set $x=x''$ with some probability when x'' is worse than the incumbent x (similar to Metropolis criterion, e.g. used in Basin Hopping and Simulated Annealing → "descent-ascent" approach)
- Set $k=k_{min}$ instead of $k=1$, set $k=k_{step}$ instead of $k=k+1$ if no better solution is found
  - Choosing large k_{min} and k_{step} implies exploration, the opposite implies exploitation (assuming that neighborhoods are nested)

VARIABLE NEIGHBORHOOD SEARCH

POSSIBLE IMPROVEMENTS

- Initialization
- Inventory of neighborhood
- Distribution of neighborhoods
- Ancillary tests
- Use of memory
- Parallel VNS
- Hybridization
- Solutions with bounds on the objective function
- Using exact algorithms for mixed-integer programming

VARIABLE NEIGHBORHOOD SEARCH

APPLICATIONS OF VNS (AND VARIANTS THEREOF) REPORTED IN THE LITERATURE

- Many industrial problems, e.g. offshore pipeline network design, cable layout, topological design of communication/distribution networks, supply chain planning, quadratic assignment, chemistry

- Combinatorial optimization problems on graphs, e.g. degree constrained spanning tree, cliques, max-cut, median cycle, and vertex coloring

- Discrete combinatorial optimization problems (unrelated to graphs), e.g. linear ordering, bin packing, and multi-dimensional knapsack

- Timetabling and manpower organization problems, e.g. team orienteering, exam proximity, design of balanced MBA student teams, and apportioning of the European Parliament

- Cluster analysis, continuous time-constrained optimization problems

- Vehicle routing problems, location-routing problem with non-linear costs

- Machine scheduling

VARIABLE NEIGHBORHOOD SEARCH

CONCLUSIONS

- Local search heuristics build upon the definition of a suitable neighborhood (set of basic moves to apply to tentative solutions).

- Unfortunately, it is difficult to know a priori which neighborhood will produce the best results. In addition, configurations which are locally optimal for one neighborhood can be further improved by a different neighborhood.

- VNS organizes the neighborhoods as sets of different strengths (amount of perturbation), it uses the simplest neighborhood at the beginning, until a local minimum is reached. At this point, neighborhoods leading to bigger perturbations are tried.

- VNS adapts the neighborhood in a reactive fashion to the characteristics of a specific problem instance and to the local situation along the search trajectory to escape locally optimal points.

VARIABLE NEIGHBORHOOD SEARCH

CONCLUSIONS

VNS has many desirable features of a metaheuristic:

- Simple and largely applicable
- Coherent: steps follow naturally from principles, not a hybrid
- Efficient and effective: very good solution quality in moderate CPU time
- Robust: performs well for various problems
- User friendly: easy to understand and implement
- Innovative: allows development of variations/extensions

VARIABLE NEIGHBORHOOD SEARCH

PRACTICAL CONSIDERATIONS

- Important to familiarize with the problem at hand, test VNS on some instances
- May be important to define a good initial solution, e.g. through a constructive heuristic
- Shaking and local search methods: either off-the-shelf or custom (based on domain knowledge)
- **FirstImprovement** of **BestImprovement**? Rule of thumb: if your initial solution is chosen at random, use **FirstImprovement**, but if a constructive heuristic is used, use **BestImprovement**.

VARIABLE NEIGHBORHOOD SEARCH

ADDENDUM: VNS vs TS

- VNS can be seen as an instance of the so-called ***exchange metaheuristics***, i.e., methods that extend the basic scheme of heuristics/local search methods by modifying their elements (the starting solution, the neighborhood, or the selection rule) in order to proceed with the search after reaching a locally optimal solution.

- As seen, VNS restarts the search from a new solution generated with a shaking procedure on the best known solution.

- Another possibility is the **Tabu Search (TS)** heuristic, that prolongs the search beyond local optima by looking for the minimum-cost neighbor solution that respects suitable tabu conditions, designed to avoid a cyclic behavior.

VARIABLE NEIGHBORHOOD SEARCH

FURTHER READING

- Hansen, Pierre, Nenad Mladenović, and José A. Moreno Pérez. "Variable neighbourhood search: methods and applications." Annals of Operations Research 175.1 (2010): 367-407.

- Mladenović, Nenad, and Pierre Hansen. "Variable neighborhood search." Computers & operations research 24.11 (1997): 1097-1100.

Questions?