Artificial Intelligence Methods (G52AIM)

Dr Rong Qu

rxq@cs.nott.ac.uk

Variable Neighborhood Search

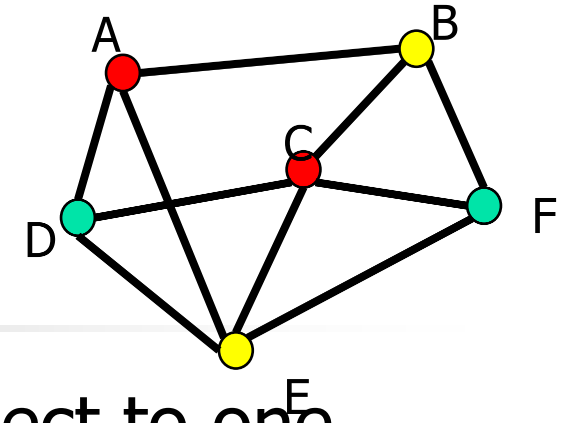
Idea of VNS

- So far on our local search based approaches, only one neighbourhood is employed

- To escape from local optimum
  - In SA, move to worse neighbourhoods base on a probability using the cooling schedule
  - In TS, move to worse neighbourhoods which are not tabued

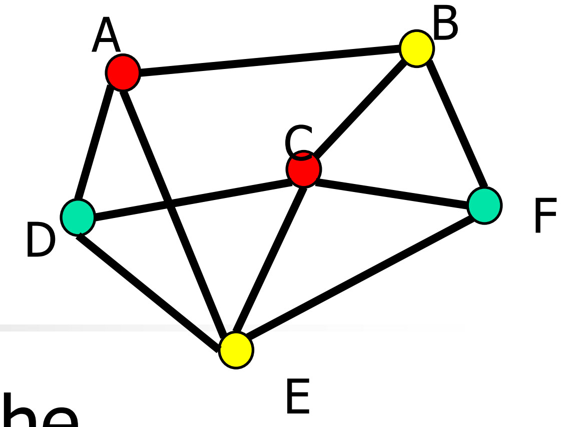
  - Any other way of escape from local optimum?

Idea of VNS



- **Fact 1.** A local minimum with respect to one neighbourhood structure is not necessary so for another;
- **Fact 2.** A global minimum is a local minimum with respect to all possible neighbourhood structures;
- **Fact 3.** For many problems local minima with respect to one or several neighbourhoods are relatively close to each other.

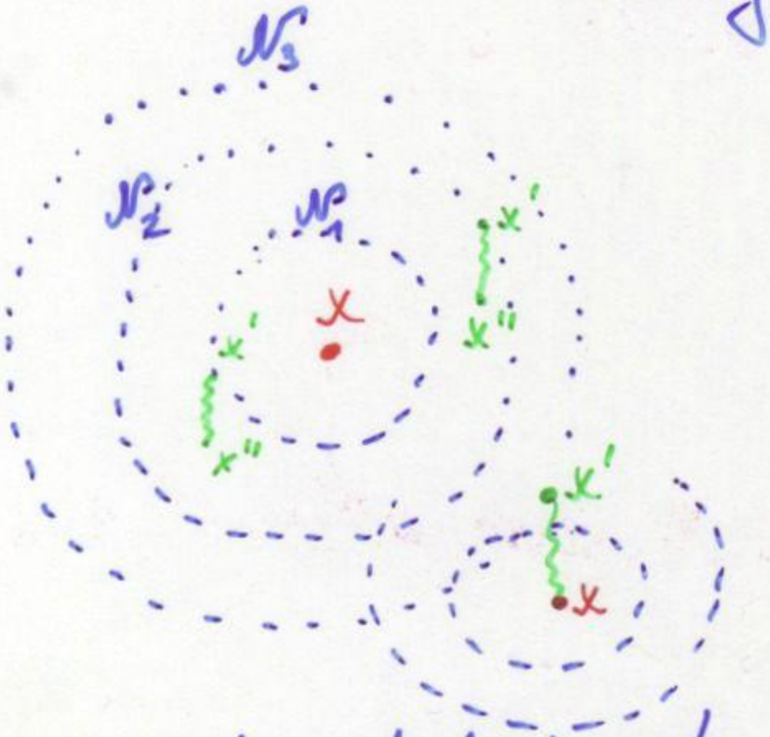
Idea of VNS



- **Principle**: systematically change the neighbourhood during the search

- Notation
  - N_k, $k = 1, 2, \ldots k_{max}$, is the set of neighbourhood structures
  - $N_k(s)$ is the set of solutions in the k^{th} neighbourhood of incumbent solution s

Basic idea: • Escape from local optima trap by changing the neighborhood structure.



• Only global opt. is local for any $\mathcal{N}_k$

Basic VNS

• Select a set of neighborhoods $\mathcal{N} = \{\mathcal{N}_1, \ldots, \mathcal{N}_{k_{max}}\}$

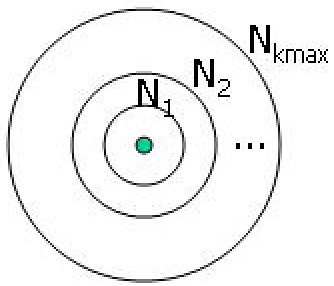
```
• Do while stop
    $k = 1$
    Do while ($k < k_{max}$)
        • Shaking
        • Localsearch
        • Move or not
```

Variable Neighborhood Search

Initialisation

Select the set of neighbourhood structures N_k

Find an initial solution x



Variable Neighborhood Search

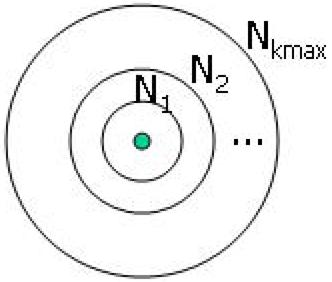
Initialisation

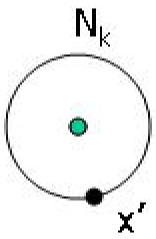
Select the set of neighbourhood structures N_k

Find an initial solution x

Repeat until stopping condition is met

- Set K=1





Variable Neighborhood Search

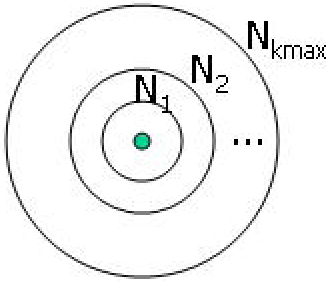
Initialisation

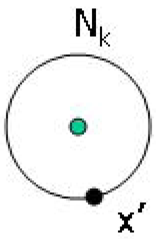
Select the set of neighbourhood structures N_k

Find an initial solution x

Repeat until stopping condition is met

- Set K=1
- **Repeat** until $k=k_{max}$
  1. *Shaking*: Generate a random point X' in $N_k(x)$



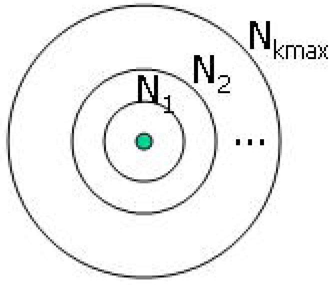


Variable Neighborhood Search

Initialisation

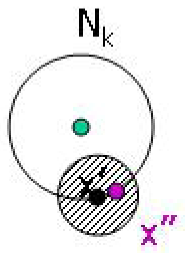
Select the set of neighbourhood structures N_k

Find an initial solution x



Repeat until stopping condition is met

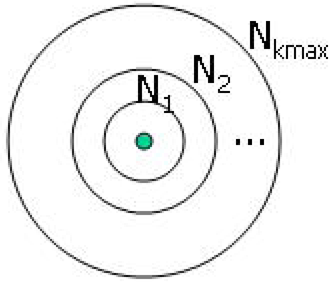
- Set K=1
- **Repeat** until $k=k_{max}$
  1. *Shaking*: Generate a random point X' in $N_k(x)$
  2. *Local Search*: x" is the obtained optimum

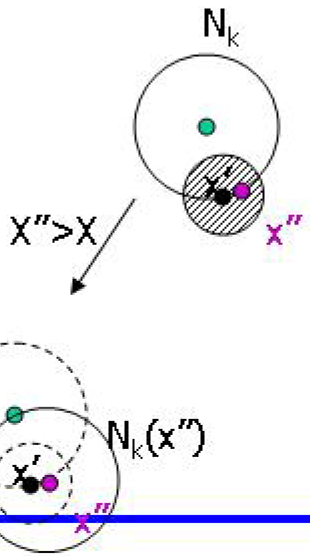


Variable Neighborhood Search

```
Initialisation
      Select the set of neighbourhood structures $N_k$
      Find an initial solution x

Repeat until stopping condition is met
      –   Set K=1
      –   Repeat until $k=k_{max}$
            1.  Shaking: Generate a random point X' in $N_k(x)$
            2.  Local Search: x'' is the obtained optimum
            3.  Move or not:
                  –   If x'' is better than x then x=x'' and k=1
```



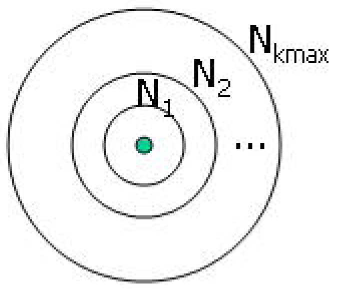


Variable Neighborhood Search

Initialisation

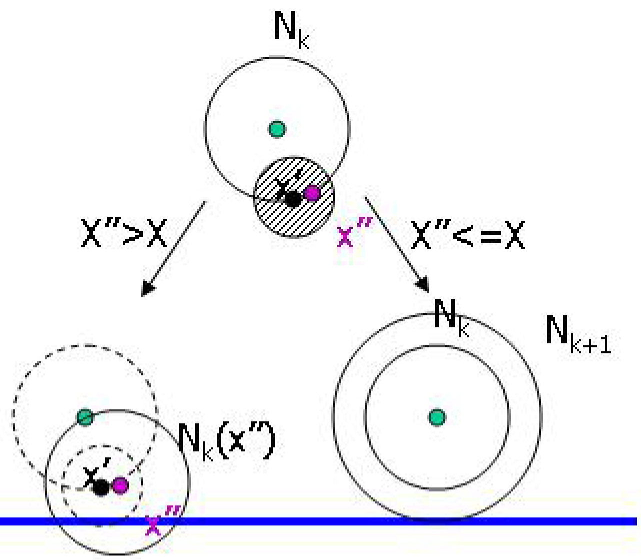
Select the set of neighbourhood structures N_k

Find an initial solution x



Repeat until stopping condition is met

- Set K=1
- **Repeat** until $k=k_{max}$
  1. *Shaking*: Generate a random point X' in $N_k(x)$
  2. *Local Search*: x'' is the obtained optimum
  3. Move or not:
     - If x'' is better than x then x=x'' and k=1
     - Otherwise k=k+1



Basic VNS

```
Procedure Basic VNS
    Select the neighbourhood set; find an initial solution, choose a stopping condition
    Repeat until stopping condition is met:
        set the first neighborhood
        Repeat until the last neighbourhood
            Shaking
            Local Search
            Move or Not
End
```

Basic VNS

- Shaking
  - Generate a point *x′* at random from the *kth* neighbourhood of *x*
- Local Search
  - Apply some local search method with *x′* as initial solution
  - *x″* be the obtained local search
- Move or Not
  - If *x″* is better than *x*, move to *x″*, go to the first neighbourhood
  - Otherwise go to the next neighbourhood

There was a typo in the original slide, i.e. x' instead of x. It should be x here, storing the incumbent solution during the search.

Basic VNS

```
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:
      $k \leftarrow 1$
      Repeat until $k = k_{max}$
        $x' \leftarrow$ RandomSolution$(N_k(x))$
        $x'' \leftarrow$ LocalSearch$(x')$
        if $f(x'') < f(x)$   then
          $x \leftarrow x''$
          $k \leftarrow 1$
        else
          $k \leftarrow k + 1$
    until stopping condition
End
```

Basic VNS – variants

- Order of the neighbourhoods
  - forward VNS: start with $k = 1$ and increase k by one if no better solution is found; otherwise set $k \leftarrow 1$
  - backward VNS: start with $k = k_{max}$ and decrease k by one if no better solution is found
  - extended version: parameters k_{min} and k_{step}; set $k \leftarrow k_{min}$ and increase k by k_{step} if no better solution is found

Variable Neighbourhood Descent

```
Procedure VND
    Select {$N_k$}, $k = 1, …, k_{max}$, find an initial solution $x$
    $k \leftarrow 1$
    Repeat until $k > k_{max}$
        $x' \leftarrow$ FindBestNeighbour$(x)$
        if $f(x') < f(x)$ then
            $x \leftarrow x'$
            $k \leftarrow 1$
        else
            $k \leftarrow k + 1$
End
```

Variable Neighbourhood Descent

- Change the neighbourhood in a deterministic way
- Final solution is locally optimal w.r.t. all neighbourhoods
- First improvement may be applied instead of best improvement
- Typically, order neighbourhoods from smallest to largest

Reduced VNS

```
Procedure Reduced VNS
    Select {$N_k$}, $k$ = 1, …, $k_{max}$, find an initial solution $x$ , choose
      a stopping condition
    $k \leftarrow 1$
    Repeat until $k = k_{max}$
        $x' \leftarrow$ RandomSolution$(N_k (x))$
        if $f(x') < f(x)$ then
            $x \leftarrow x'$
            $k \leftarrow 1$
        else
            $k \leftarrow k + 1$
End
```

Reduced VNS

- Same as basic VNS except that no LocalSearch procedure is applied
- Only explores randomly different neighbourhoods
- Can be faster than standard local search algorithms for reaching good quality solutions

Decisions in VNS

- Number and type of neighbourhoods to be used
- Order of their use in the search
- Strategy for changing the neighbourhoods
- Local search methods
- Stopping condition

- There is no need to design sophisticated acceptance criteria to escape from local optima
- Design of neighbourhoods presents to be crucial for VNS

Decisions in VNS – for TSP

- Number and type of neighbourhoods to be used
- Order of their use in the search
- Strategy for changing the neighbourhoods
- Local search methods
- Stopping condition

Reading Materials

- N. Mladenovic, Variable neighbourhood search. Invited seminar, City University, London, March 8, 2007.
- P. Hansen and N. Mladenovic, Variable neighborhood search: Principles and applications, EJOR 43, 2001.

Learning Objectives

- VNS Basics
  - Basic VNS
  - Reduced VNS
  - Supporting idea
  - Decisions in algorithm design
- Be able to implement VNS in your coursework
  - Making appropriate decisions to design effective and efficient VNS for the problem in hand