DM865 – Spring 2019
Heuristics and Approximation Algorithms

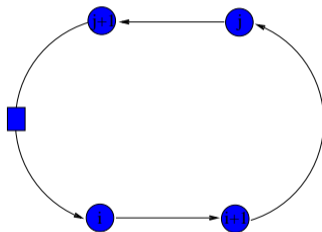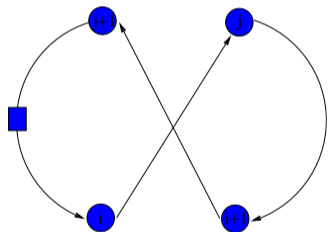# Local Search for
Traveling Salesman Problem

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Intra-route Neighborhoods

2-opt

$$\{i, i+1\}\{j, j+1\} \longrightarrow \{i,j\}\{i+1, j+1\}$$
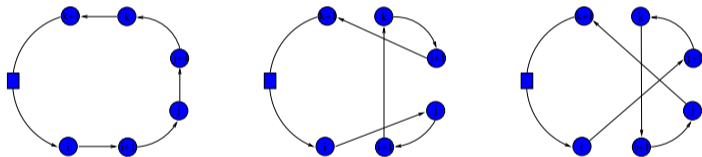

$O(n^2)$ possible exchanges
One path is reversed

Intra-route Neighborhoods

3-opt

$$\{i, i+1\}\{j, j+1\}\{k, k+1\} \longrightarrow \dots$$

$O(n^3)$ possible exchanges
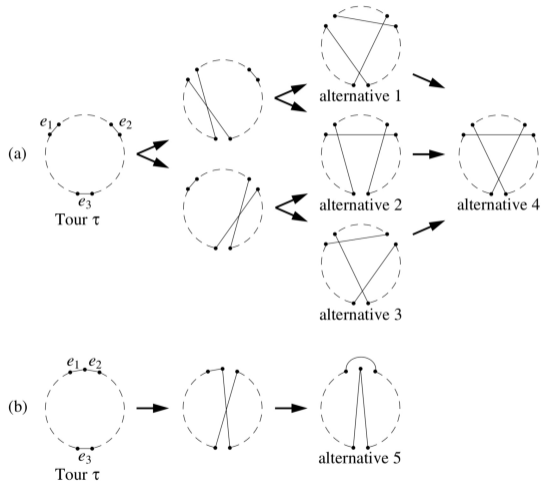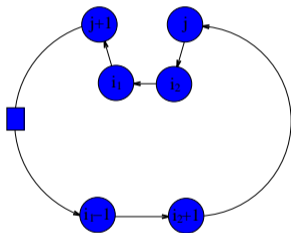Paths can be reversed

Possible 3-Exchanges

Figure from [MAK]

Intra-route Neighborhoods

Or-opt [Or (1976)]
$$\{i_1 - 1, i_1\}\{i_2, i_2 + 1\}\{j, j + 1\} \longrightarrow \{i_1 - 1, i_2 + 1\}\{j, i_1\}\{i_2, j + 1\}$$

sequences of one, two, three consecutive vertices relocated
$O(n^2)$ possible exchanges — No paths reversed

Table 17.1 Cases for k-opt moves.

k	No. of Cases
2	1
3	4
4	20
5	148
6	1,358
7	15,104
8	198,144
9	2,998,656
10	51,290,496

[Appelgate Bixby, Chvátal, Cook, 2006]

Local Search Example

Random-order first improvement for the TSP

- **Given:** TSP instance G with vertices $v_1, v_2, \ldots, v_n$.
- **Search space:** Hamiltonian cycles in G;
- **Neighborhood relation N:** standard 2-exchange neighborhood
- **Initialization:**
  search position := fixed canonical tour $< v_1, v_2, \ldots, v_n, v_1 >$
  "mask" P := random permutation of $\{1, 2, \ldots, n\}$
- **Search steps:** examine neighbors in order of P (does not change throughout search)
  evaluate neighbors w.r.t. cost of tour $f(s)$
  accept the first improvement
- **Termination:** when no improving search step possible
  (local minimum)

Local Search Example

Iterative Improvement for TSP

TSP-2opt-first(s)
input: *an initial candidate tour $s \in S(\in)$*
output: *a local optimum $s \in S_\pi$*

for $i = 1$ to $n - 1$ **do**
    for $j = i + 1$ to n **do**
        if $P[i] + 1 \geq n$ or $P[j] + 1 \geq n$ **then** *continue* ;
        if $P[i] + 1 = P[j]$ or $P[j] + 1 = P[i]$ **then** *continue* ;
        $\Delta_{ij} = d(\pi_{P[i]}, \pi_{P[j]}) + d(\pi_{P[i]+1}, \pi_{P[j]+1}) + $
                $-d(\pi_{P[i]}, \pi_{P[i]+1}) - d(\pi_{P[j]}, \pi_{P[j]+1})$
        if $\Delta_{ij} < 0$ **then**
          `UpdateTour(s,P[i],P[j])`

is it really?

Local Search Example

Iterative Improvement for TSP

TSP-2opt-first(s)
input: *an initial candidate tour $s \in S(\in)$*
output: *a local optimum $s \in S_\pi$*

FoundImprovement:=$TRUE$;
while FoundImprovement **do**
    FoundImprovement:=$FALSE$;
    for $i = 1$ to $n - 1$ **do**
        for $j = i + 1$ to n **do**
            if $P[i] + 1 \geq n$ or $P[j] + 1 \geq n$ **then** *continue* ;
            if $P[i] + 1 = P[j]$ or $P[j] + 1 = P[i]$ **then** *continue* ;
$$\Delta_{ij} = d(\pi_{P[i]}, \pi_{P[j]}) + d(\pi_{P[i]+1}, \pi_{P[j]+1}) + \\ -d(\pi_{P[i]}, \pi_{P[i]+1}) - d(\pi_{P[j]}, \pi_{P[j]+1})$$
            if $\Delta_{ij} < 0$ **then**
                `UpdateTour(s,P[i],P[j])`
                FoundImprovement=$TRUE$

Local Search Example

Efficient implementations of 2-opt, 2H-opt and 3-opt local search.

A. Neighborhood pruning (exact or heuristic)
   Fixed radius search + Candidate lists + DLB

B. Delta evaluation (already in $O(1)$)

C. Data structures

Details at black board and references [Bentley 92,Johnson McGeoch 2002, Appelgate Bixby, Chvátal, Cook, 2006]

Local Search for TSP

- *k*-exchange heuristics
  - 2-opt
  - 2.5-opt
  - Or-opt
  - 3-opt
- complex neighborhoods
  - Lin-Kernighan
  - Helsgaun's Lin-Kernighan
  - Dynasearch
  - ejection chains approach

Implementations exploit speed-up techniques
- A. neighborhood pruning:
  - fixed radius nearest neighborhood search
  - neighborhood lists: restrict exchanges to most interesting candidates
  - don't look bits: focus local search to "interesting" part
- B. delta evaluation
- C. sophisticated data structures

Implementation examples by Stützle: http://www.sls-book.net/implementations.html

TSP data structures

Tour representation:

- determine pos of v in π
- determine succ and prec
- check whether u_k is visited between u_i and u_j
- execute a k-exchange (reversal)

Possible choices:

- $|V| < 1.000$ array for π and π^{-1}
- $|V| < 1.000.000$ two level tree
- $|V| > 1.000.000$ splay tree

Moreover static data structure:

- priority lists
- k-d trees

Look at implementation of local search for TSP by T. Stützle:

File: http://www.imada.sdu.dk/~marco/DM811/Resources/ls.c

```
two_opt_b(tour); % best improvement , no speedup
two_opt_f(tour); % first improvement , no speedup
two_opt_best(tour); % first improvement with dlbs ,
                    % fixed radius near neighbour searches , neughbourhood lists
two_opt_first(tour); % best improvement with dlbs ,
                     % fixed radius near neighbour searches , neughbourhood lists
three_opt_first(tour); % first improvement
```

Table 17.2 Computer-generated source code for *k*-opt moves.

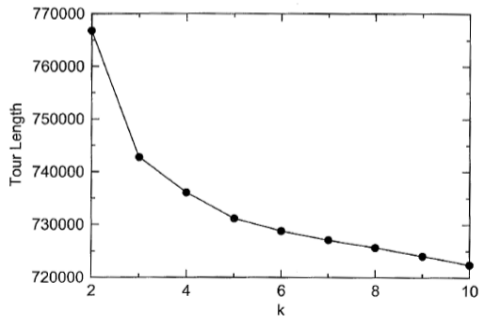| *k*	No. of Lines
6	120,228
7	1,259,863
8	17,919,296

Figure 17.1 *k*-opt on a 10,000-city Euclidean TSP.