Tutorial November 19, 2002

Very Large Scale Neighborhood Search

Ravi Ahuja
James Orlin

Neighborhood search

- *Combinatorial Optimization*: minimize (f(x) : x ∈ X)
- f is typically linear,
- X is finite

- *Neighborhood Function*:
- For each x ∈ X, there is a *neighborhood* N(x);
- We say that x is a *local optimum* if f(x) ≤ f(y) for all y ∈ N(x);

- *Neighborhood Search (local improvement algorithm)*
- *begin*
  - initialize with some x ∈ X;
  - *while* x is not a local optimum *do*
    - replace x by some y ∈ N(x) such that f(y) < f(x);
- *end*

Very Large Scale Nbhd (VLSN) search?

- **Rule of Thumb for Larger Neighborhoods**:
  - improved local optima
  - greater search time
- **This talk**:
  - Focuses on **VERY LARGE** neighborhoods that can be searched very efficiently (preferably in polynomial time) or are searched heuristically.
  - often exponentially large neighborhoods

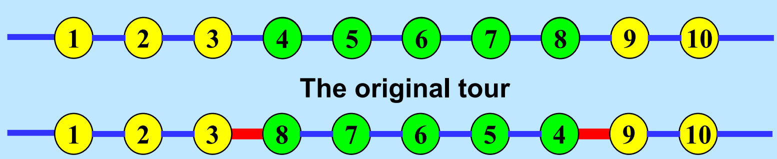
Some References

- See Aarts and Lenstra [1997]
  Local Search in Combinatorial Optimization
- "A Survey of Very Large Scale Neighborhood Search Techniques", Ahuja, Ergun, Orlin, and Punnen [1999]
  http://web.mit.edu/jorlin/www/
- "A study of exponential neighborhoods...." Deineko and Woeginger [2000]
- Two sessions at this conference (plus this tutorial).

Overview of this talk

- Part 1
  - Introduction
  - Domination Number
  - The Extended Neighborhood
  - Recursively Defined Neighborhoods
- Part 2
  - Application to Fleet Routing
  - Application to Partitioning

Classical Nbhd -- 2-opt



A 2-neighbor of the original tour

We say that a tour T' is a ***2-neighbor*** of a tour T if it is possible to obtain T' from T by adding two edges and deleting two edges.

We call the nbhd the ***2-opt neigbhorhood***

T' = T + (3,8) + (4,9) - (3,4) - (8,9).
Obtained by the operation Flip[4,8]

Assume the current tour is 1, 2, 3, ..., n

Summary on this part of the talk

- **Domination number: an alternative to usual worst case analysis which may offer different insights**
- **The extended neighborhood. An alternative to size of the neighborhood.**

I'm planning on using the 2-opt neighborhood. It has n^2 neighbors.

Pretty good. But please check out larger neighborhoods.

Domination Number

- Introduced by Glover and Punnen (1996)
- Suppose that N is a neighborhood function for an instance I = minimize (f(S) : S $\in$ F)
- Suppose x* is worst local optimum in F
- *Domination Number* for N
  Dom(N, I) = | { y : f(y) > f(x*) |
- The domination number for 2-opt is the number of solutions that are guaranteed to be worse than every 2-opt solution.

- Theorem. Rubilinekki 1971, Sarvanov 1971. For an asymmetric TSP, each 2-opt solution has objective value at least as good as half the tours, that is, (n-2)! tours.
- Corollary. The domination number for 2-opt on ATSP with n cities is at least (n-2)!
- Punnen, Margot, Kabadi, 2001. A version of 2-opt obtains a solution with domination number at least (n-2)! in $O(n^3 \log n)$ steps.

- **Theorem**. Rubilinekki 1971, Sarvanov 1971. For an asymmetric TSP, each 2-opt solution has objective value at least as good as half the tours, that is, (n-2)! tours.
- **Corollary**. The domination number for 2-opt on ATSP with n cities is at least (n-2)!
- **Punnen, Margot, Kabadi, 2001.** A version of 2-opt obtains a solution with domination number at least (n-2)! in $O(n^3 \log n)$ steps.

The neighborhood seems very large to me. Every local optima dominates (n-2)! solutions.

Great! That seems large to me too.

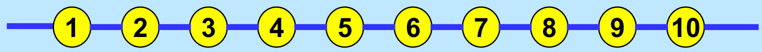
More on Domination Number

- The domination number for 2-opt is better than for the domination number for Christofides heuristic
- The domination number for 2-opt is better than for the assignment heuristic, which has (n/2)! solutions in the nbhd.
- See Punnen and Kabadi [2002]
- Gutten and Yeo [1998 and 1999]

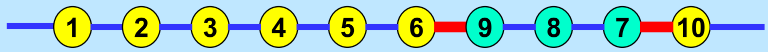
Motivation for Domination Number

- An interesting new metric, which may provide insights on which neighborhood is better in practice.
- Possibly less sensitive to data perturbation
- Perhaps more effective than the size of a neighborhood
- Next: extended neighborhood

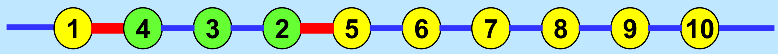
Independent 2-exchanges



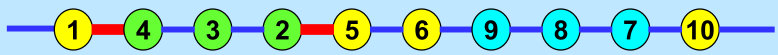
The original tour T



Flip[7,9] and obtain T_1



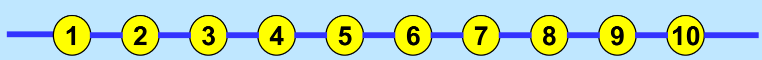
Flip[2,4] and obtain T_2



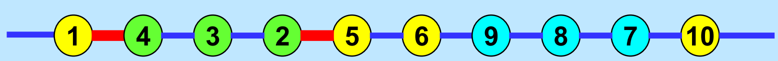
Flip[2,4] and Flip[7, 9]

Two exchanges, Flip[i,j] and Flip[i',j'] are **independent** if $i' > j+1$ (or $i > j'+1$).

More on independent 2-opt



The original tour T



Flip[2,4] and Flip[7,9]

A collection of flips is ***independent*** if every two of them are independent.

One can search the independent 2-opt nbhd in $O(n^2)$ time. (Potts and van de Velde [1995])

The size of the independent 2-opt nbhd is $\Omega(1.75^n)$.

Other aspects of independent 2-opt

- **2-opt is in many ways similar to Independent 2-opt.**
  - same set of local optima
    - same complexity for the problem of finding a local optimum
  - same worst case bound
  - same domination number
  - any local search using 2-opt is also a local search using independent 2-opt
  - empirically, they behave similarly

I'm searching the independent 2-opt neighborhood. Its size is at least 1.75^n

This is great!

Can you do better?

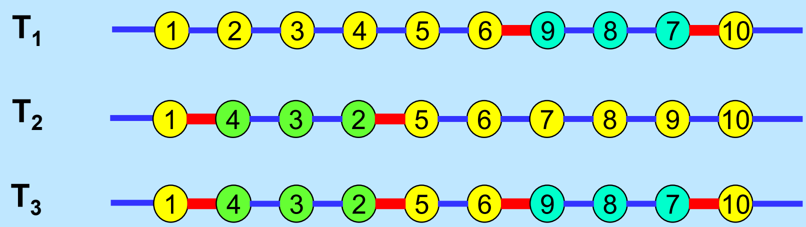
2-opt vs independent 2-opt:

- The neighborhoods really are equivalent in some sense (to be defined soon)
- So.... maybe 2-opt can be viewed as acting like a neighborhood of exponential size.
- Perhaps this explains in part why 2-opt is a very good neighborhood for searching.

LO-equivalence O. and Sharma [2002]

- We say that two neighborhoods N and N' are ***LO-equivalent*** (Locally Optimal) for a combinatorial optimization problem if
  - for any instance I, the set of locally optimal solutions for I are the same for both N and N',
- The neighborhoods are different, but the locally optimal solutions are the same

On LO-equivalence



Let N be the 2-opt neighborhood

Let N' be the independent 2-opt neighborhood

Theorem. The 2-opt neighborhood and the independent 2-opt neighborhood are LO-equivalent

The extended neighborhood

- **If N and N' are LO-equivalent, then they are both LO-equivalent to N", where**
  $N''(x) = N(x) \cup N'(x)$

- **The largest neighborhood that is LO-equivalent to N is called the *extended neighborhood* of N.**
  We denote it as *N*.*

- **A new metric:**
  - **the size of the extended neighborhood**

On Extended Neigbhorhoods

- If a neighborhood is ***exact***, then the extended neighborhood is the set of all feasible solutions.

- Let N be the neighborhood
  – We can determine if $y \in N^*(x)$ by solving a single linear program
  – Relies on a polyhedral characterization of the extended neighborhood

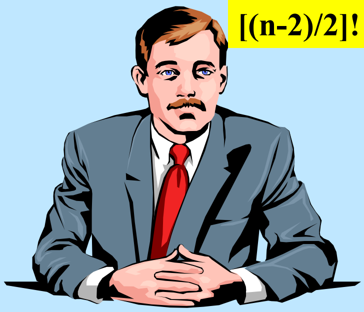
On the extended neighborhood 2-opt*

- The size of 2-opt* is $>$ [(n-2)/2]!
  – Perhaps this helps to explain why 2-opt obtains such excellent locally optimal solutions.

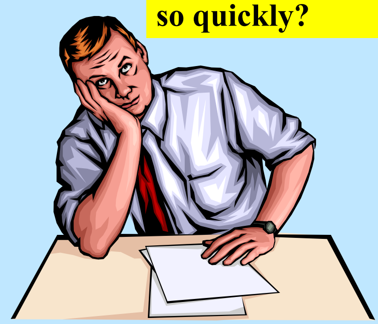
- One can recognize whether $y \in$ 2-opt*(x) by solving a linear program.

- Finding the best solution in 2-opt* is NP-hard.

I'm searching 2-opt* . It size is at least [(n-2)/2]!



How did you manage to search such a large neighborhood so quickly?



More on extended neighborhoods

- Consider the extended neighborhood for the swap neighborhood for the QAP. It is the same, that is, N* = N.

- Next: using DPs in VLSN search.

Using DP to search neighborhoods based on research of Ergun and Orlin [2002]

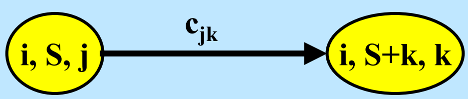
- Some heuristics use variants of DP to search neighborhoods
  - Deineko and Woeginger [2000]
  - Sarvanov and Doroshko [1981]
  - Potts and van de Velde [1995]
  - Balas and Simonides, and more
- Standard approach: define the nbhd first
- Our approaches
  - Solve the DP first
  - Create the neighborhood and the DP in parallel

Concurrent approach

- Representing the neighborhoods searched in terms of recursions
  - Use the same recursion to form a new DP
  - In our examples, the neighborhoods are exponential, but the DP has a polynomial number of states.

DP-recursions (as a shortest path)

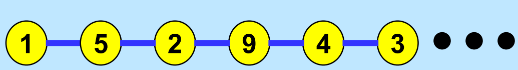
- **States: <i, S, j>**
  - e.g., <3, {3, 4, 5, 6}, 6> represents a subtour that starts at node 3, passes through 4, 5, and ends at 6.
  - {j} will denote the subtour of one node; it is shorthand for <j, {j}, j>
- **There is an arc from <i, S, j> to <i, S+k, k> with cost c_{jk}**



Usual DP recursion is equivalent to finding a shortest path from {1} to <1, N, j> for each j.

Illustration using usual DP for TSP

- **Let F denote the set of all subtours with initial node 1**
- **Recursive description of F**
  - {1} $\in$ F
  - If A $\in$ F, and k $\notin$ A, then A, k $\in$ F

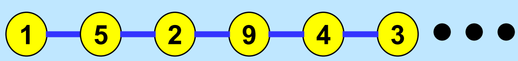


The same recursion for the state space

- **Recursive description of F**
  - **–{1} ∈ F**
  - **–If A ∈ F, and k ∉ A, then A, k ∈ F**

- **Recursive description of state space**
  - **– {1} ∈ N**
  - **– If <i, S, j> ∈ N and if k ∉ S, then <i, S+k, k> ∈ N and the cost of the “move” is c_{jk}**

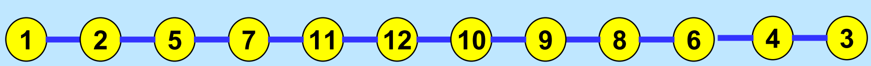
<1, {1, 2, 4, 5, 9}, 4>



<1, {1, 2,3, 4, 5, 9}, 3>

Recursive nbhd: pyramidal neighbors

In a *pyramidal tour*, the node labels increase to n and then decrease.



There is an straightforward DP recursion that finds the best pyramidal tour.

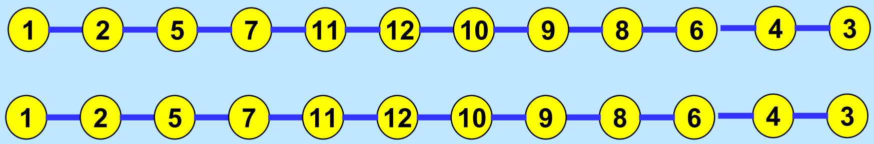
Let T be any tour. Relabel the cities so that T = 1, 2, 3, …, n. A *pyramidal neighbor* of T is any pyramidal tour under this relabeling.

Next: how to find pyramidal tours using recursion

Creating the pyramidal neighborhood

Let F be a collection of subtours.

Complete tours in F will be the pyramidal neighbors.



Rule 1: {n} ∈ F
Rule 2: If A ∈ F, and if j+1 = min(A), then (j, A) ∈ F and (A, j) ∈ F.

These rules define the pyramidal neighborhood

The pyramidal neighbor state graph

Rule 1: {n} ∈ F
Rule 2: If A ∈ F, and if j+1 = min(A), then (j, A) ∈ F and (A, j) ∈ F.

Rule 1': {n} ∈ N
Rule 2' :
Suppose <i, S, k> ∈ N, and j+1 = min(S).
Then <i, S+j, j> ∈ N and <j, S+j, k> ∈ N.

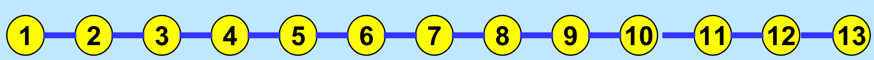
$O(n^2)$ states reachable from {n}, and $O(n^2)$ possible transitions between these states.

On recursively defined neighborhoods

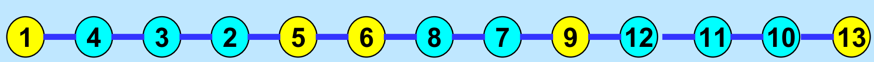
- Simple and natural.

- Potential for
  - simpler proofs of correctness
  - simpler analysis
  - potential for automation
  - potential for new VLSN

Independent 2-opt Neighborhood



The original tour T



An independent 2-opt neighbor

Rule 1: $\{1\} \in F$

Rule 2: If $A \in F$, and if $i-1 = \max(A)$, then
$A, \{j, j-1, \ldots, i\}, j+1 \in F$.

The independent 2-opt state graph.

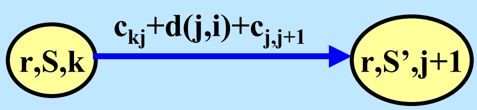
Rule 1: $\{1\} \in F$

Rule 2: If $A \in F$, and if $i-1 = \max(A)$, then
$A, \{j, j-1, \ldots, i\}, j+1 \in F$.

Let $d(j,i)$ be the cost of the subtour $j, j-1, \ldots i$

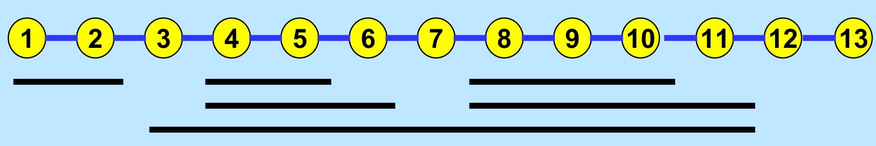
Rule 1': $\{1\} \in N$

Rule 2': Suppose that $<r, S, k> \in N$, and
$i-1 = \max(A)$. Let $S' = S \cup \{i, i+1, \ldots, j, j+1\}$
$<r, S', j+1> \in N$.



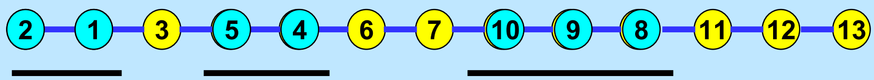
$O(n^2)$ states reachable from $\{1\}$, and $O(n^2)$ possible transitions between these states.

Twisted Neighborhood



Create a sequence of intervals on [1, n], such that for any two of them, one contains the other or they are non-overlapping.

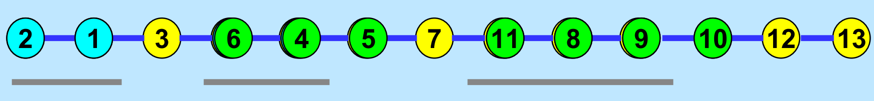
Twisted Neighborhood



Create a sequence of intervals on [1, n], such that for any two of them, one contains the other or they are non-overlapping.

Flip intervals of cities in order of containment

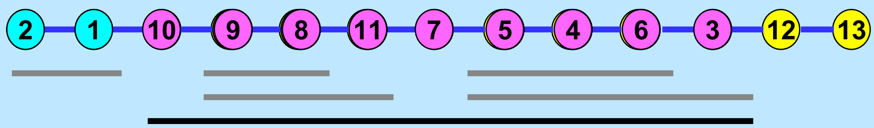
Twisted Neighborhood



Create a sequence of intervals on [1, n], such that for any two of them, one contains the other or they are non-overlapping.

Flip intervals of cities in order of containment

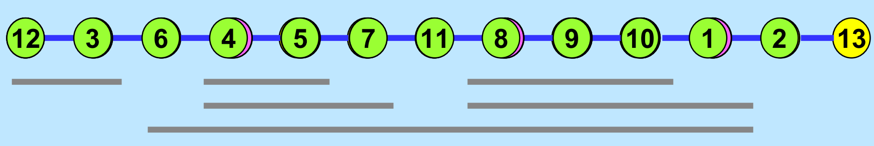
Twisted Neighborhood



Create a sequence of intervals on [1, n], such that for any two of them, one contains the other or they are non-overlapping.

Flip intervals of cities in order of containment

Twisted Neighborhood

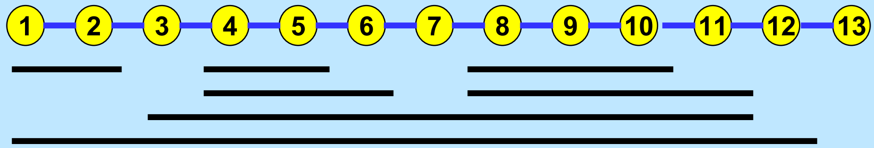


Create a sequence of intervals on [1, n], such that for any two of them, one contains the other or they are non-overlapping.

Flip intervals of cities in order of containment

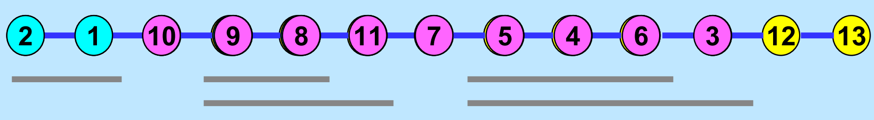
The twisted neighborhood is somewhat complex to describe, and leads to complex moves.

Twisted Neighborhood using recursion



Rule 1: {j} ∈ F, for each j = 1 to n
Rule 2: Suppose A ∈ F, B ∈ F and
max (A) = min(B) – 1; then
(A,B) ∈ F, and rev(A,B) ∈ F

Illustration of Recursion



Rule 1: {j} ∈ F, for each j = 1 to n
Rule 2: Suppose A ∈ F, B ∈ F and
max (A) = min(B) – 1; then
(A,B) ∈ F, and rev(A,B) ∈ F

3 + 6 4 5 = 3 6 4 5

3 6 4 5 + 7 = 3 6 4 5 7

Rev 3 6 4 5 7 + 11 8 9 10

Summary of Recursive Neighborhoods

- Recursive Neighborhoods Unifies much of VLSN search
- Induce DP recursions
- New Conceptualization can lead to new neighborhoods as well.
- Automates much of the solution approach

- Next: Ravi Ahuja will provide three applications of VLSN search.