Neighborhood Search Made Difficult: Very Large Scale Neighborhood Search

By Ravidra Ahuja
Ozlem Ergun
James Orlin
and Abraham Punnen

Overview of the Talk

- **Introductory Material**
  - **Introductory notations and definitions**
  - **A way of describing neighborhood search techniques**
  - **Neighborhood / Search / Select**
  - **Very Large Scale Neighborhood Search (VLSN Search)**
- **Variable depth methods and ejection chains**
- **Network flow based approaches**
  - **Negative cycle based methods**
  - **Shortest path based methods**
  - **Assignment based methods**
- **Neighborhoods based on polynomially solvable special cases of the problem**
- **Summary and future directions**

Heuristics

Rules of thumb or procedures that may get a good solution, but are not guaranteed to come up with the best solution.

Construction heuristics. **(In terms of TSP). Create a tour, possibly one link at a time.**

Example. The nearest neighbor heuristic. Start at some node and always visit the next closest unvisited node.

Improvement heuristics: **(In terms of TSP). Start with a tour, and try to make minor modifications in the tour to improve its length.**

Neighborhood search is a particular way of implementing improvement heuristics

Terminology for Neighborhood Search

***Combinatorial Optimization*: minimize (f(S) : S ∈ F)**

- **f is typically linear,**
- **F is finite**

***Neighborhood Function*:**

- **For each S ∈ F, there is a *neighborhood* N(S);**
- **We say that S is a *local optimum* if f(S) ≤ f(T) for all T ∈ N(S);**

Neighborhood Search

begin

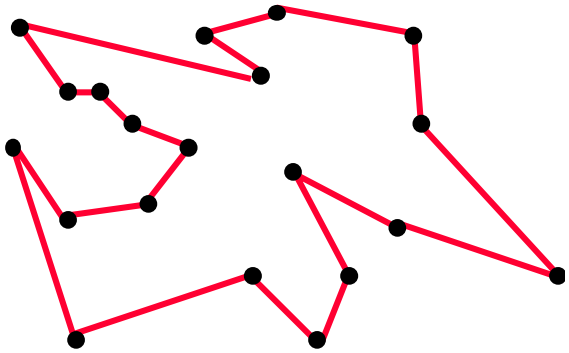
initialize with some S ∈ F;

while* S is not a local optimum *do

replace S by some T ∈ N(S) such that f(T) < f(S);

end

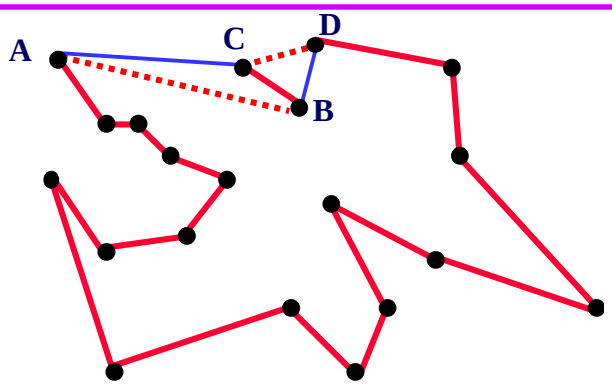
Traveling Salesman Problem



What is a minimum length tour that visits each point?

THE classic combinatorial optimization problem.

2-Exchanges



One can delete AB and CD and add AC and BD and get a shorter length 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.

2-opt for TSP

Neighborhood Search:

begin

start with some tour T ∈ F

while there is a 2-neighbor T' of T with a lower objective value, replace T by T'

end

A solution is *2-optimum* if there is no 2-neighbor that has a lower length. A solution is locally optimum if there is no neighbor with an improved objective.

Easily extended to 3-opt or to K-opt

Some Extensions of Neighborhood Search

- ***Simulated Annealing***
- ***Tabu Search***
- **Neighborhoods based on a pair of solutions**
  - ***crossovers* in genetic algorithms**
  - ***scatter search***
- ***Variable depth search and ejection chains***

These methods draw on the idea of a "neighborhood" but differ from neighborhood search in other ways.

Accordingly, we need concepts beyond "the neighborhood."

A Neighborhood Search Technique has Three Parts

Neighborhood/Search/Select

1. ***A neighborhood structure*: represented conceptually as a *neighborhood graph*. Example: for TSP, the nodes of the neighborhood graph are tours, and two tours are adjacent if they are 2-neighbors of each other.**
2. ***A method for searching the neighborhood graph*. Typically, there is an incumbent, and neighbors are selected. In simulated annealing, a neighbor is selected at random. In VLSN search, the neighbors are searched using a special purpose algorithm.**
3. ***A method for selecting the next incumbent*. Typically, this is done by selecting the best solution from the search. In simulated annealing randomness is used. In Tabu search, the best solution satisfying the tabu restrictions is selected unless …**

A Remark About the Size of Neighborhoods

Observations:

- **The larger the neighborhood, the lower the cost of local optima (on average)**
- **The larger the neighborhood, the greater is the time requirement to find an improved neighbor**
- ***Goal*: Balance the two conflicting objectives**

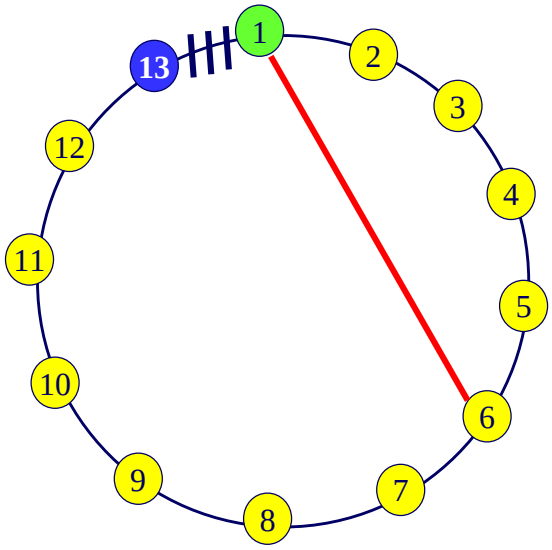
Our survey:

Focuses on VERY LARGE neighborhoods that can be searched very efficiently (preferably in polynomial time.) or are partially searched heuristically.

Two types of VLSN algorithms that we address.

- **Variable depth methods: the neighborhood graph has small degree, but the method scans neighborhoods of large distance from the incumbent.**
- **Algorithms in which the size of the neighborhood is exponential (and which the neighborhood is searched in polynomial time).**

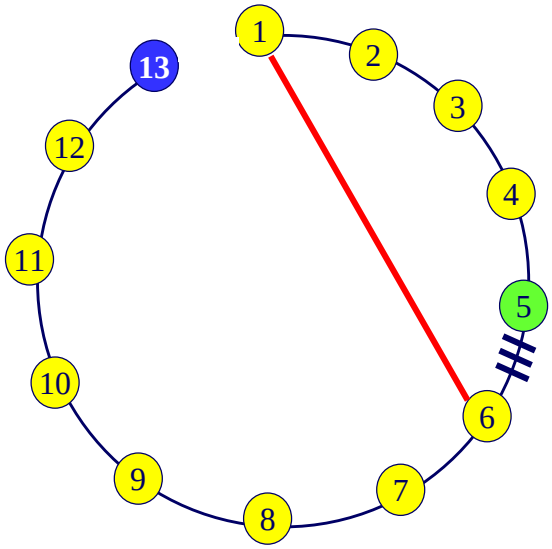
The Lin-Kernighan Algorithm: A Variable Depth Method



Choose some edge to delete say (1,n), creating a path.

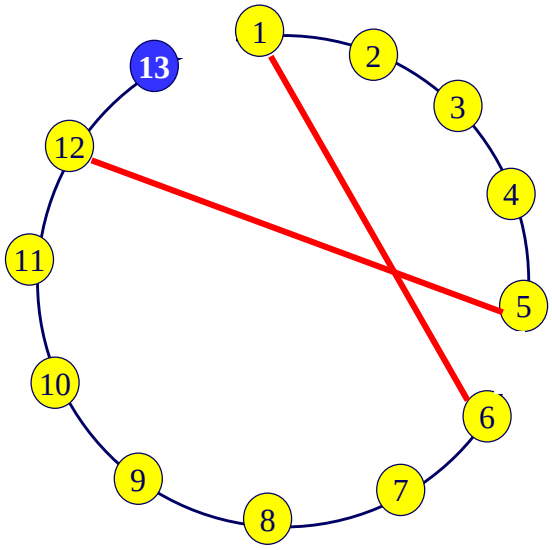
Add in the edge with minimum length. Creating a *"stem and cycle."*

Successively subtract and add arcs



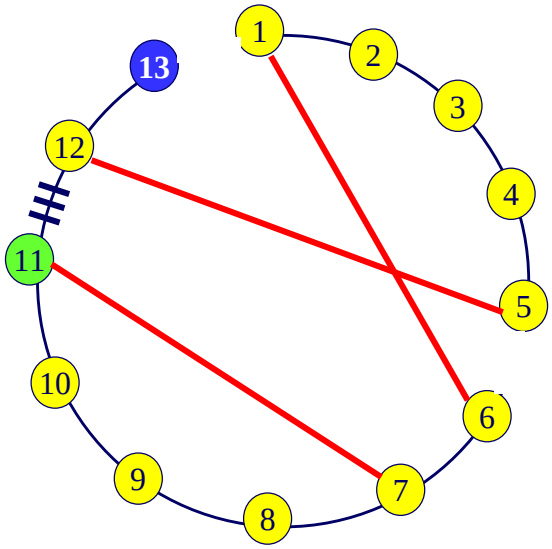
Delete an edge to recreate a path.

Successively subtract and add arcs



Add in the edge with minimum length. Creating a *stem and cycle*.

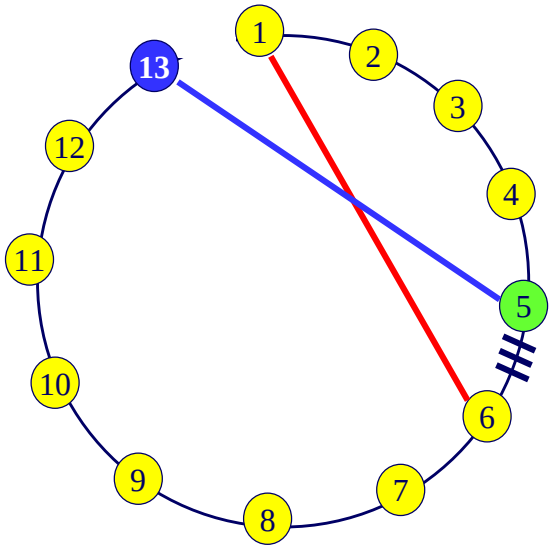
Successively subtract and add arcs



Keep deleting edges to create paths, and adding min length edges to create a *stem and cycle*.

Do not delete red edges. Stop if deleting plus adding increases the cost.

Creating Cycles from Paths



Each path can be completed to a cycle. Choose the best of these cycles as the next incumbent.

Summary of Lin-Kernighan's technique

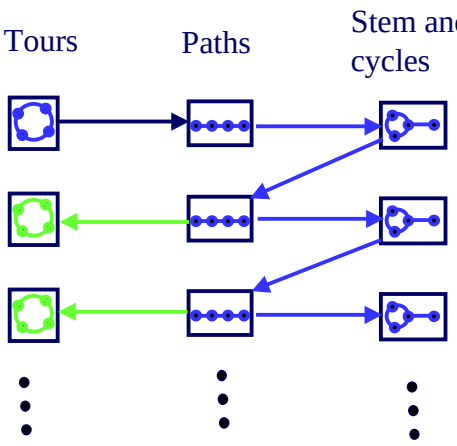
Procedure Variable_Depth_Search
begin
$S_1 := T_1 := S$
for j = 2 **to** 2n **do**
$(T_j, S_j) = Move(S, T_{j-1})$
select the set S_k **that minimizes** $(f(S_j): j = 1 \text{ to } k)$;
end.

Move: either creates a path T_j **(and a cycle** S_j**)**
or it creates a stem and cycle T_j

"stem and cycle" and paths are the *reference structures.*

Alternating paths like this are often referred to as *ejection chains.*

The Lin-Kernighan Approach (Neighborhood graph)



This neighborhood graph is a representation of adjacent "reference structures." Each move of L-K is a search of this neighborhood. The best tour is selected.

More on Variable Depth Methods

There are alternative ways of defining the neighborhood for each problem.

- **L-K deletes an edge and then adds an edge.**
- **One could add and then delete.**
- **It really is quite different in practice.**

One can search large subsets of the neighborhoods more efficiently.

One can employ additional heuristics in the search, or in the selection.

It can be applied to a variety of combinatorial problems, especially ones in which feasibility is not really the issue.

Some References

Variable Depth Search

- **Kernighan and Lin [1970], Lin and Kernighan [1973]**

Ejection Chains.

- **Glover [1996]**
- **Dondorf and Pesch [1994]**
- **Laguna, Kelley, Gonzales-Velarde, and Glover**
- **Pesch and Glover [1997]**
- **Punnen and Glover [1996]**
- **Rego [1998]**
- **Rego and Roucairol [1996]**

Network Flow Based Techniques

Neighborhood Search

begin

initialize with some S ∈ F;

while **S is not a local optimum** ***do***

replace S by some T ∈ N(S) such that f(T) < f(S);

end

In network flow based techniques, the neighborhood is searched implicitly by solving a network flow problem on an "improvement graph."

Type 1: Cycle Canceling Algorithms

Illustration on Minimum Cost Flows: start with a feasible flow x* for the minimum cost flow problem.

1. Create the residual network G(x*). [The improvement graph]

2. Look for a negative cost cycle in G(x*); if none exist, you have a local (and global) optimum solution.

3. Send flow around the cycle, and update x*.

Remark: This can be generalized to a heuristic for integer multicommodity flows -- Schneur and Orlin [1997] , Rockafellar [1984]

Next: a detailed example, applied to Partitioning Problems

VLSN for Partitioning Problems

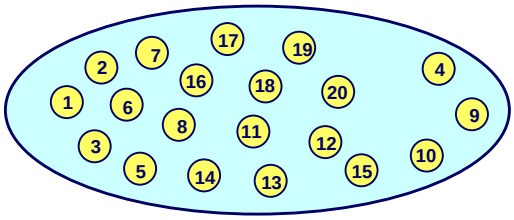
- S : set of objects
- Partition S into subsets (S_1, S_2, S_3, .. , S_p)
- Define an optimization function $f(S_i)$ for each subset S_i
- Find the partition S_1, S_2, … , S_p such that

$$\sum_{p=1,K} f(S_p)$$

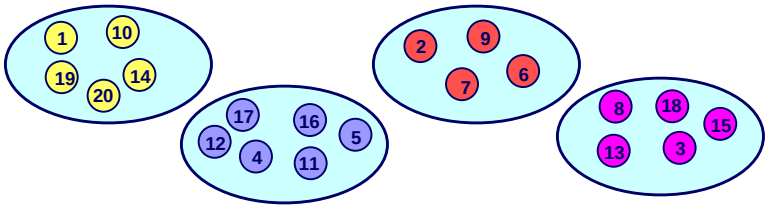
is minimum.

Partitioning Problems

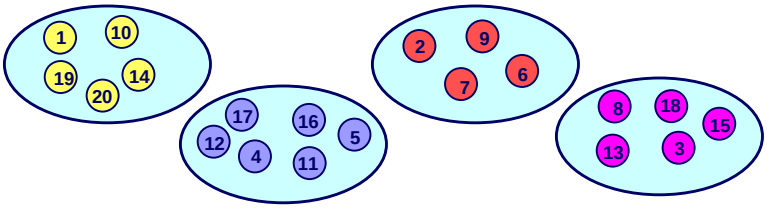
Given a set S of 20 objects



Partition the set into four subsets



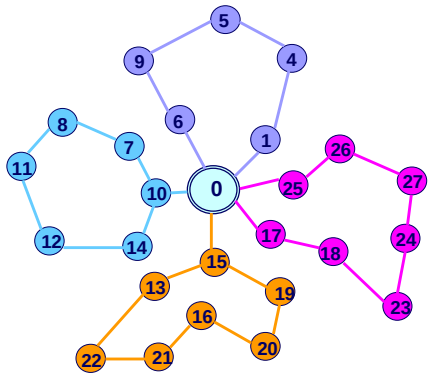
The Cost Structure



The cost for part S_i is $f(S_i)$.

Find the partition with minimum total cost.

Vehicle Routing Problems, Scheduling Problems, Clustering Problems, and more



Vehicle Routing: Assign trucks to pick up cargoes so as to minimize total travel time and meet service requirements.

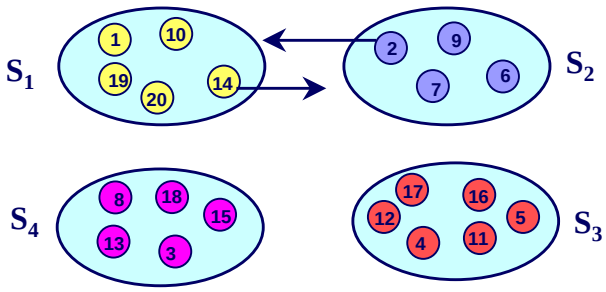
Note: $f(S_i)$ may be difficult to compute for a subset S_i.

Partitioning Problems are Difficult to Solve

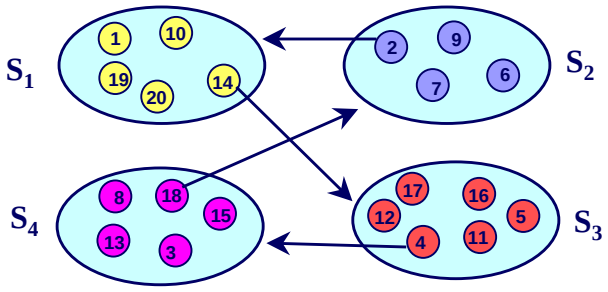
Most partitioning problems are NP-complete.

They can be quite difficult to solve in practice.

A simple neighborhood search technique: 2-exchanges



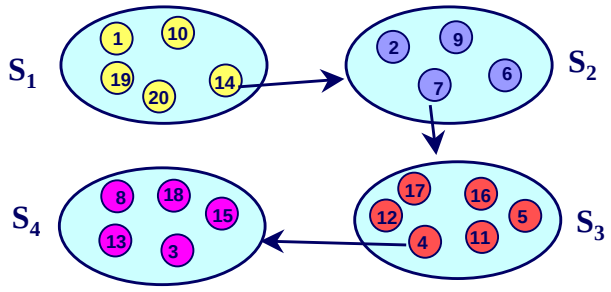
The Cyclic Exchange Neighborhood (Multi-Swap)



Each subset either (i) is unchanged or (ii) has one item inserted and another item deleted

The number of neighbors can grow exponentially in the number of items.

A Path Exchange



After a path exchanges one subset increases by 1 and another decreases by 1

More on Cycle Based Exchanges

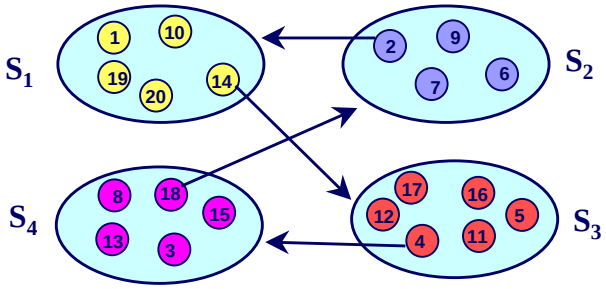
Path exchanges can be transformed to cyclic exchanges

We use the concept of improvement graph to identify profitable cyclic exchanges

Some References

- **Thompson and Orlin [1989] and**
- **Thompson and Psaraftis [1993]**
- **Thompson [1989]**
- **Talluri [1996]**
- **Ahuja, Orlin, and Sharma [1998]**
- **Gendereau, Guertin, Potvin, and Sequin [1998]**

The Effect of a Cyclic Exchange (Multi-Swap)



Create an improvement graph, with arcs (i,j) for all i and j in different subsets.

Before

$f(S_1)$
$+ f(S_2)$
$+ f(S_3)$
$+ f(S_4)$

After

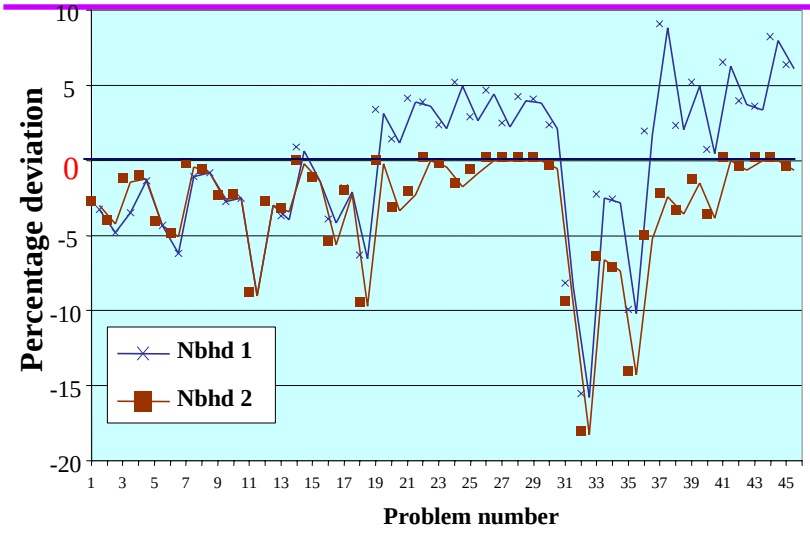
$f(S_1 + 2 - 14)$
$+ f(S_2 + 18 - 2)$
$+ f(S_3 + 14 - 4)$
$+ f(S_4 + 4 - 18)$

$c_{2,14} = f(S_1 + 2 - 14) - f(S_1)$
$c_{18,2} = f(S_2 + 18 - 2) - f(S_2)$
$c_{14,4} = f(S_3 + 14 - 4) - f(S_3)$
$c_{4,18} = f(S_4 + 4 - 18) - f(S_4)$

The Improvement Graph

- **There is a one-to-one correspondence between cyclic exchanges and "subset-disjoint" directed cycles in G.**
- **A negative cost subset-disjoint cycle in G defines an improving cyclic exchange.**
- **Identifying negative cost subset-disjoint problem is NP-complete.**
- **We can find negative cost subset-disjoint cycles effectively in practice using either heuristics or implicit enumeration.**

Data for the Capacitated Minimum Spanning Tree Problem



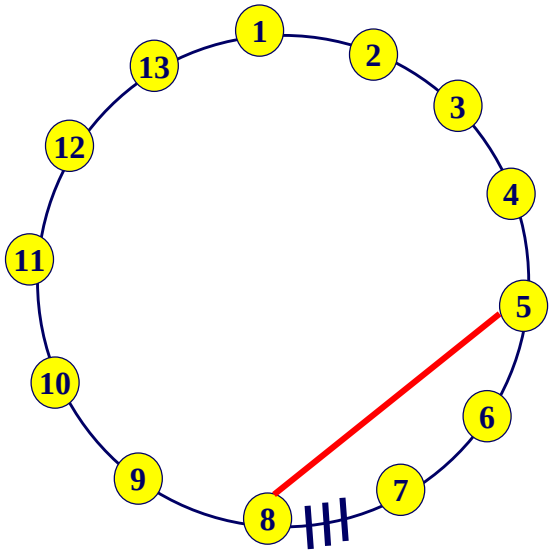
Comparison on heterogeneous demand problems.

Path Based Neighborhoods

Illustration for the Maximum Flow Problem : start with a feasible flow x* for the maximum flow problem.

1. **Create the residual network G(x*). [The improvement graph]**
2. **Look for a directed path from s to t in G(x*); if none exist, you have a local (and global) optimum solution.**
3. **Send flow in the path, and update x*.**

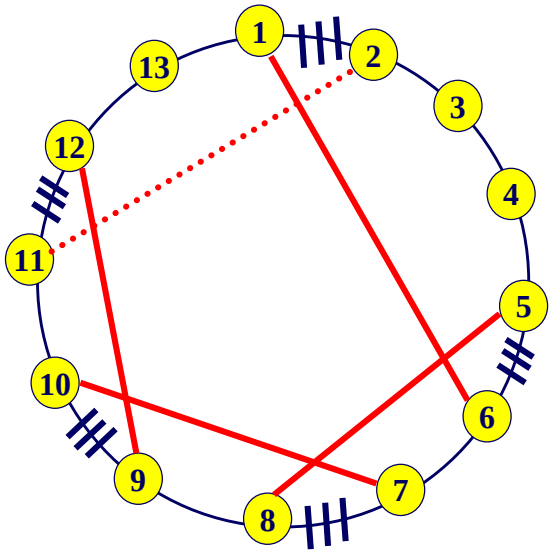
A Path (or Cycle) Based Neighborhood



Recall the L-K variable depth method.

Special rule here: if (i,j) is added, then (j-1, j) is deleted.

A Path (or Cycle) Based Neighborhood

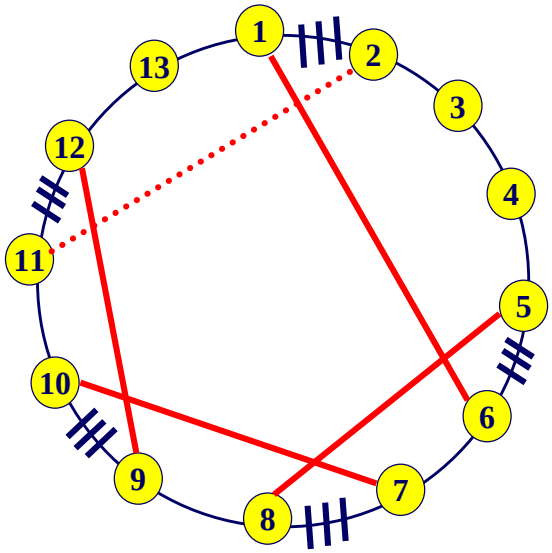


Add and delete arcs in parallel.

Let S denote the set of arcs to be added. (This determines the set to be deleted.)

When will this create a feasible tour?

A Path (or Cycle) Based Neighborhood

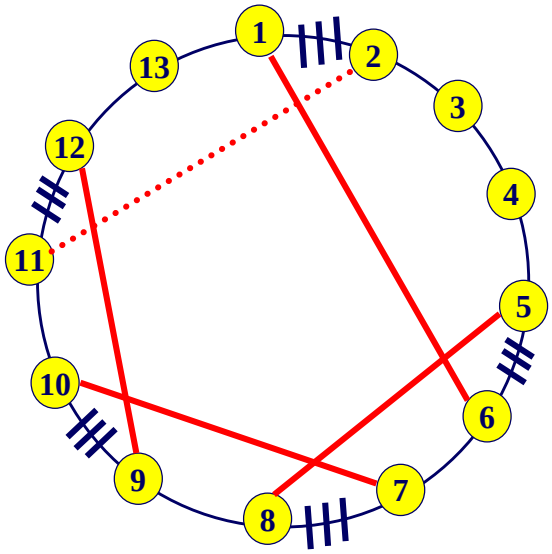


Sufficient condition: It creates a tour if the additions and deletions can be represented as a *monotone odd cycle.*

***[1,5]* means: add (1,6) and delete (5,6).**

1-5-7-9-11-1 increasing order, odd number of nodes

A Path Based Neighborhood



Improvement graph:

Arc (i,j) means that edge (i, j+1) is added to T and edge (j,j+1) is deleted.

Find the minimum cost odd monotone cycle in the improvement graph. (Shortest path problem.)

Additional Information on Path Based Search

A single connected path, or the union of disjoint paths

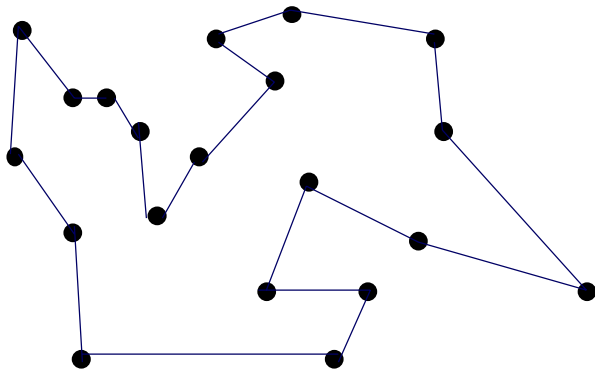
Extension to other scheduling problems

The shortest path problem can be solved very quickly.

Some References on Path Based methods and DP based methods:

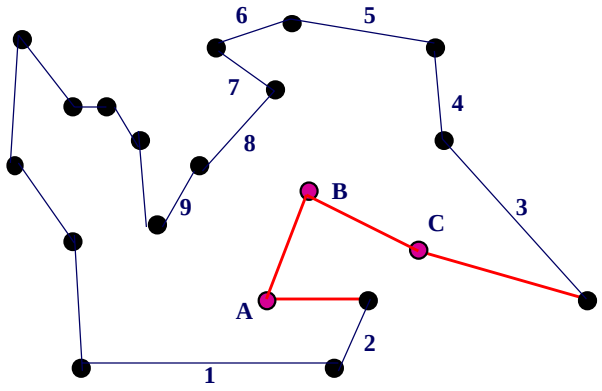
- **Carlier and Villon [1990]**
- **Congram, Potts, and Van de Velde [1999]**
- **Hurink [1998]**
- **Potts and Van de Velde [1995]**
- **Punnen and Glover [1996]**
- **Spille, Firla and Weismantel [1999]**

An Assignment Based Neighborhood



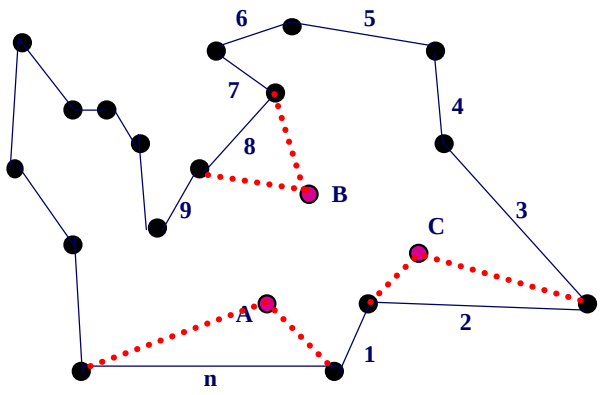
Start with a tour. Select a subset of cities on a path

The Assignment based Neighborhood



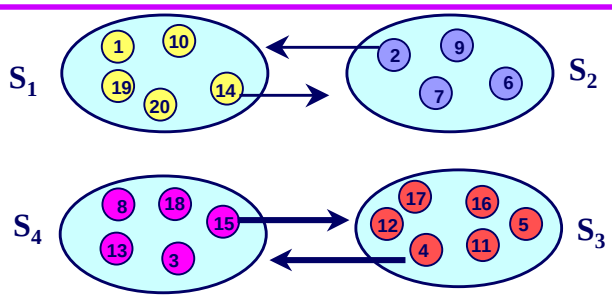
Delete the nodes of the path and reinsert them optimally.

The Assignment-Based Neighborhood



To reinsert the nodes, solve an assignment problem. The cost of (A,n) is the incremental cost of replacing the edge n by edges passing through A.

A Non-Bipartite Matching Based Neighborhood

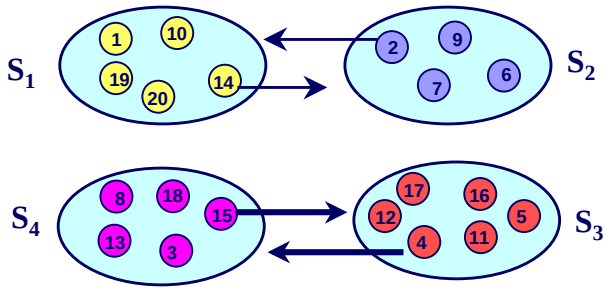


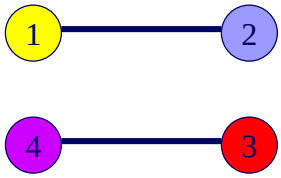
Two swaps for the partitioning problem are *independent*, if their corresponding four subsets are all disjoint.

(S_1, …, S_k) and (T_1, …, T_k) are *I-neighbors* if one can be obtained from the other by independent swaps.

Searching for the best *I-neighbor* is a non-bipartite matching problem on a related *improvement graph*.

The Improvement Graph: Matching Version





List the best arc joining S_1 to S_2.

Arc weight = improvement

More on Assignment Based Neighborhoods

One can insert nodes on a path, or nodes on multiple paths

One can add the restriction that certain nodes of the path stay together

The assignment problem can be solved very quickly. Problems with up to two million nodes have been solved optimally.

Some References.

- **Sarvanov and Doroshko [1981] (in Russian)**
- **Deineko and Woeginger [1997]**
- **Gutin [1998] and [1999]**
- **Gutin and Yeo [1999]**
- **Punnen [1999]**

Neighborhoods Based on Restrictions

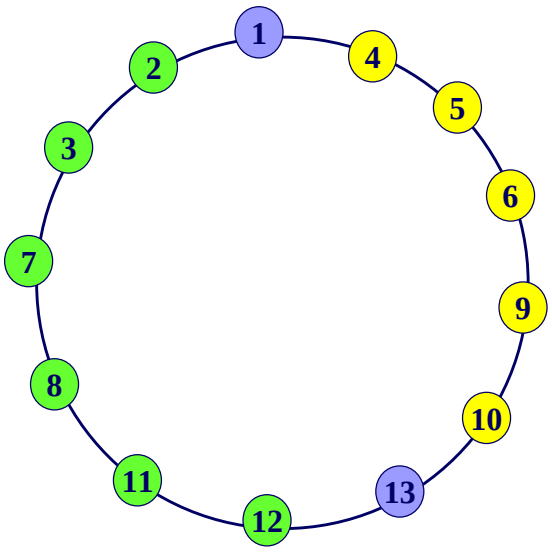
Vast literature based on polynomially solvable special cases.

Idea: turn a special case into a neighborhood.

Illustrations:

- **Halin Graphs**
- **Pyramidal Tours**

Pyramidal Tour

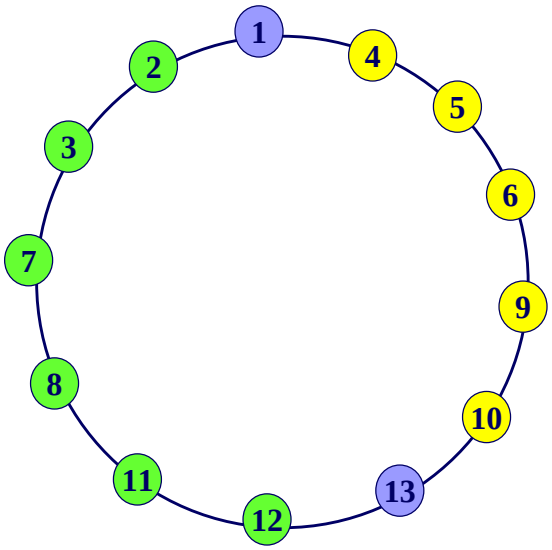


Tour is monotone increasing from node 1 to node n both clockwise and counterclockwise

Best pyramidal tour :

$O(n^2)$ time.

Pyramidal Neighbors



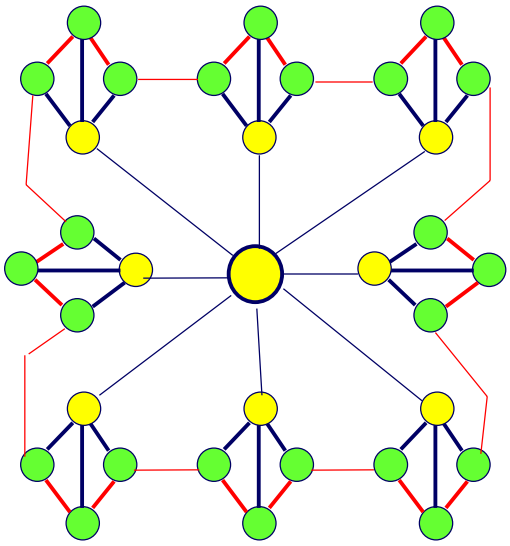
First relabel nodes of T so that they are 1 to n in order.

***Pyramidal neighbor*: a tour T' that is pyramidal after relabeling.**

Best pyramidal neighbor :

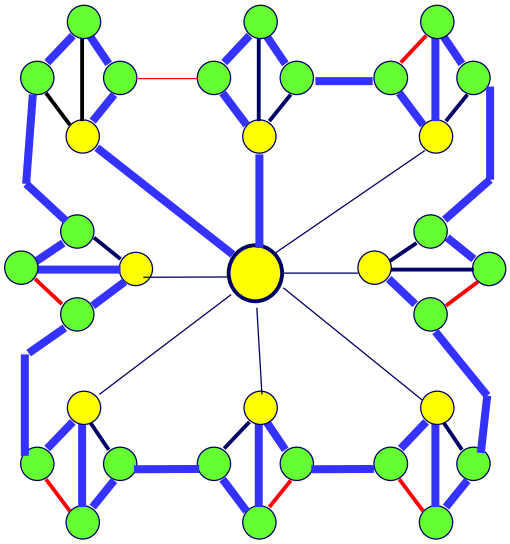
$O(n^2)$ time.

Halin Graphs



A *Halin Graph* is a tree whose nodes do not have degree 2, and the leaves are connected in the plane to form a cycle.

Halin Graphs



One can find min cost tours in *Halin Graph* in O(n) time.

Cornuejols, Naddef, and Pulleyblank [1983]

Halin Neighbors

Idea: embed a Tour inside a Halin graph, and optimize over the neighborhood.

Suggested by Punnen

Requires: a polynomial time algorithm for embedding T in a Halin Graph

General Approach

Let X be an *NP-hard* problem.
Let X' be a restriction of X, where X ∈ *P*.

Suppose there is a subroutine CreateNeighbor(S)

INPUT: feasible subset S, for (F,f) ∈ X
OUTPUT: Instance (F',f) ∈ X', with F' ⊆ F, and S ∈ F'

S is feasible for the special case F' that is solvable in polynomial time.

We refer to F' as the *X'-induced neighborhood* of S.

Summary

Several Methods for VLSN Techniques

- **Variable Depth Methods and Ejection Chains**
- **Network Flow Based Techniques**
  - **Cycle based methods**
  - **Path based methods**
  - **Assignment and matching based neighborhoods**
- **Neighborhoods Based on Polynomially Solvable Special Cases**

Future Research

- **Development of new neighborhood search techniques**
- **Empirical analysis of VLSN algorithms**
- **Theoretical analysis of VLSN algorithms**