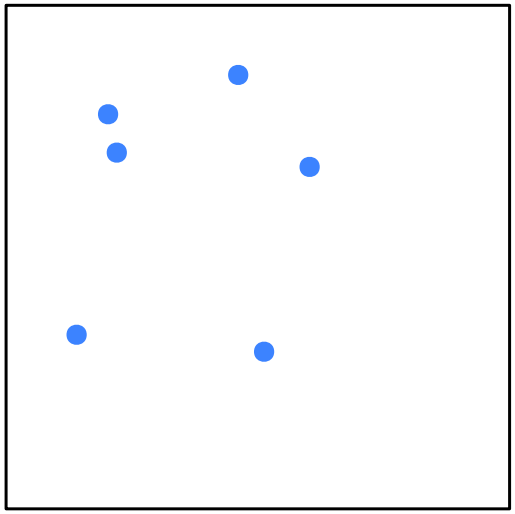
Finding An Optimal Layout for a Given Access Pattern

Nathaniel Jones

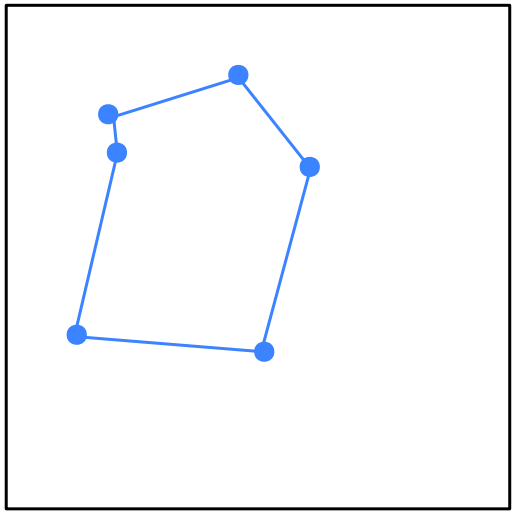
Traveling Salesman Problem

Given a set of points, find a route through the points to minimize the total distance traveled.

Given



Find



Traveling Salesman Problem

- Familiar, well-studied problem
- Many existing algorithms
- Touched on it in this class

The Reverse Problem

Given a route through a set of points (where the route can visit nodes multiple times), arrange the points such that the total distance traveled is minimized.

Given

A→B→A→C
→D→A→E→D

Find

D	C		
E	A	B	

The Reverse Problem

- Far less studied
- No known algorithms I am aware of
- In need of a catchy name

Why solve this problem?

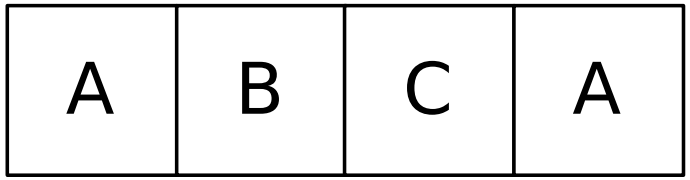
- Primary application is computer memory and instruction caches
  - Data that is closest together is fastest to load
  - Instructions that are close together are decoded together
  - Guaranteed function locality could allow for simpler instruction fetching logic

Modeling the problem

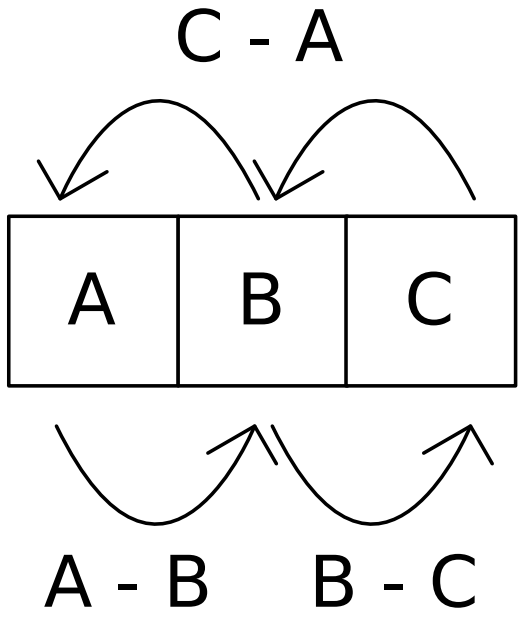
- Given an ordered list of nodes S, find a permutation P such that following the sequence of traversals in S requires the least amount of moves

Modeling the problem

Sequence

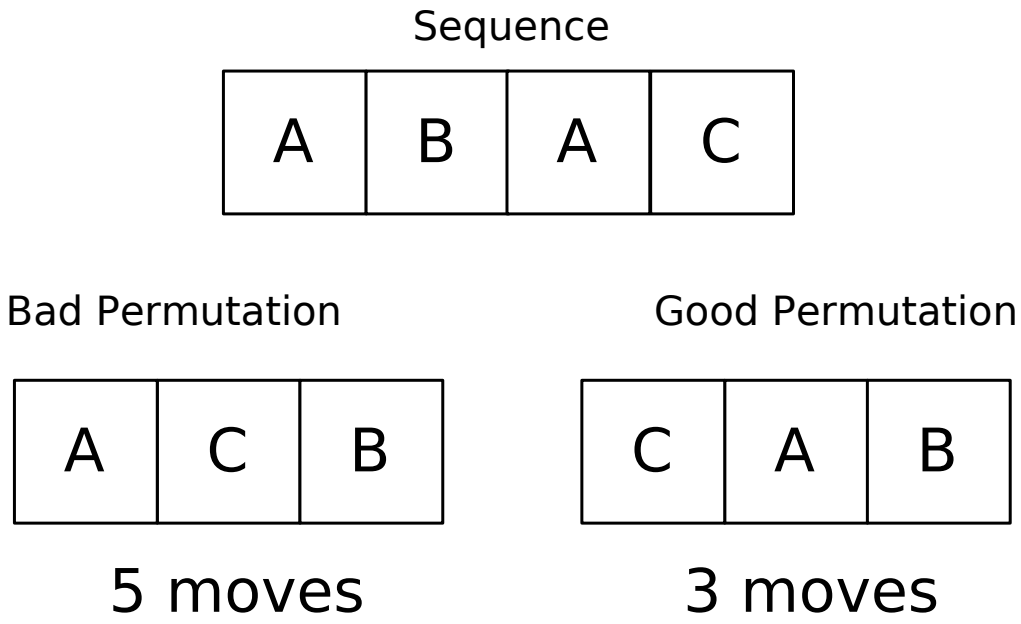


Permutation



A Useful Heuristic

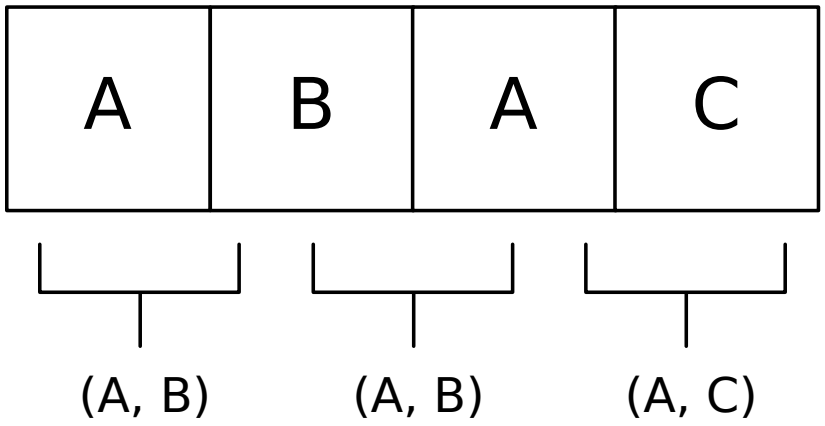
Intuitively, the nodes traversed between the most should be closest together.



How do we ensure nodes traveled between are close together?

- First, determine which nodes are most frequently traversed
  - Each pair of nodes is assigned a rank of 0 to start
  - Every time two nodes appear next to one another in the sequence *S*, increment their rank
  - Each individual node is also ranked by how many traversals it is involved in

How do we ensure nodes traveled between are close together?



Ranks

(A, B)	2
(A, C)	1
(B, C)	0

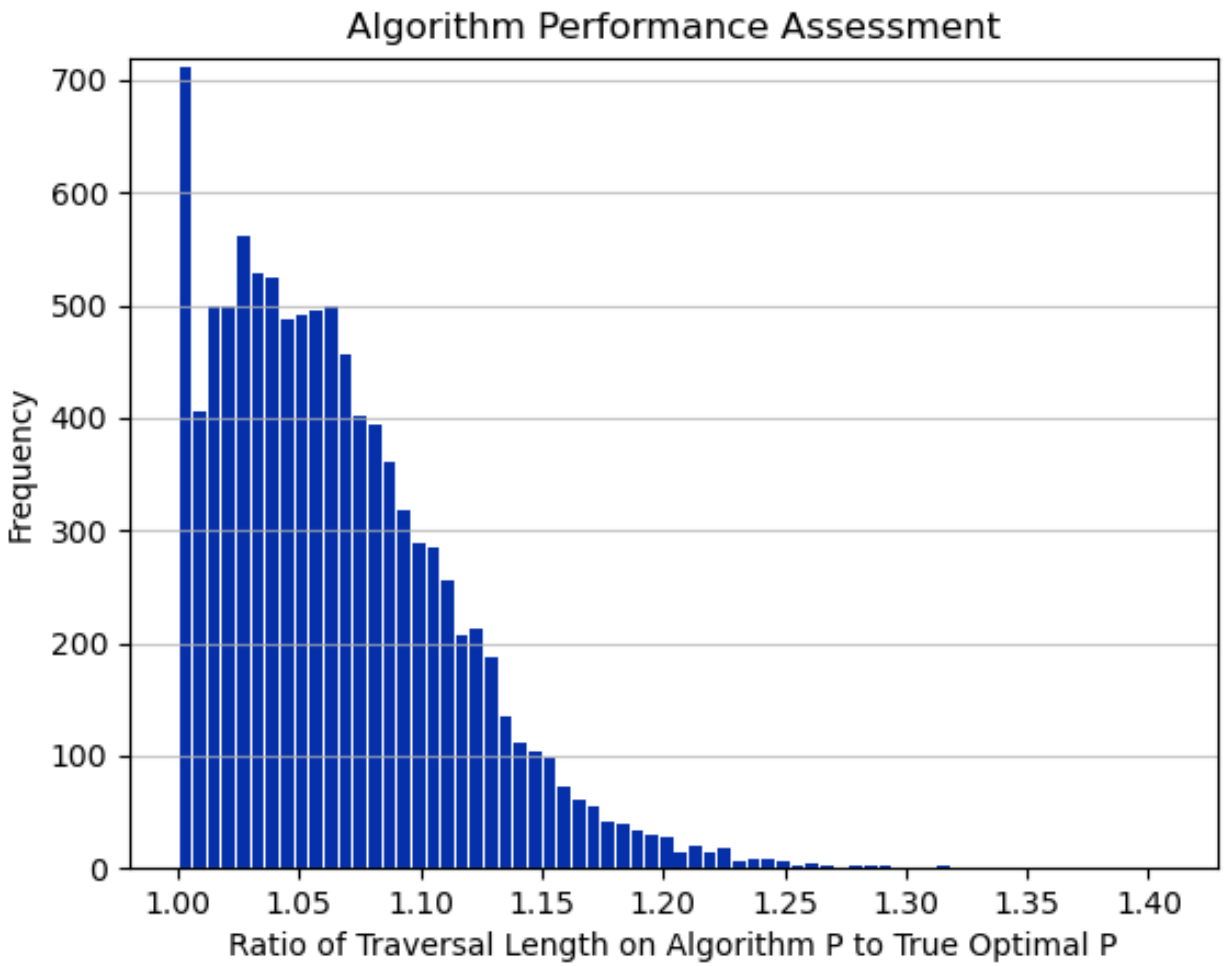
How do we ensure nodes traveled between are close together?

- Then add the node pairs to *P* from highest ranked to lowest ranked
  - Both nodes in a pair are added at the same time to ensure adjacency
  - Add nodes to *P* until all nodes have been added

Choosing where in *P* to add a pair of nodes

- Whether to prepend or append depends on the current contents of *P*
  - The pair is placed to be closest to the inserted nodes that they have the most traversals with
  - If no traversals with inserted nodes, they are placed so that the most traversed of them is on the outside

Results


Algorithm Performance Assessment
Frequency
0
100
200
300
400
500
600
700
1.00
1.05
1.10
1.15
1.20
1.25
1.30
1.35
1.40
Ratio of Traversal Length on Algorithm P to True Optimal P

Algorithm Analysis

- The algorithm runs in $O(n^2)$ with respect to the length of the sequence
- Much better than the O(n!) of brute-force searching!
- Resulting *P* is usually within 15% of the true optimal *P*

Algorithm Analysis

- Algorithm seems to struggle in some cases (long tail in results)
- Worst-case performance seems to be about +30% greater total traversal length
- More testing and measurements needed

Future Work

- Analysis of difficult cases
- Determine how the worst-case scenario changes with sequence length / node count
- Improve on the worst case scenario
- Expand to higher dimensions