Better Routing

Lecture 7 - Management Science

Dr. Tobias Vlćek

Introduction

Client Briefing: Artisan Bakery

. . .

Master Baker's Morning Dilemma:

"Every morning at 5 AM, our delivery van leaves with fresh bread for 12 cafés across the city. Our driver takes 3 hours for what should be a 2-hour route. We're burning fuel and arriving late. Can you optimize our morning delivery?"

The Delivery Challenge

Artisan Bakery's daily logistics puzzle:

- 12 Cafés: Each expecting fresh bread by 8 AM
- One Van: Limited capacity, must visit all locations
- Time Windows: 3 cafés open early (6:30 AM) and need priority
- Current Problem: Driver uses "gut feeling" for routing
- Cost Impact: Extra hour = €50 fuel + €30 labor + unhappy customers

. . .

> ! Important
>
> The Stakes: Poor routing costs €80+ daily, or €29,200 annually. Plus reputation damage from late deliveries!

Quick Recap: Greedy Decisions

Last week we learned greedy algorithms for scheduling:

- SPT: Process shortest jobs first
- EDD: Process by earliest due date
- Fast & Simple: Made quick decisions, no looking back

. . .

Question: Can we use the same greedy approach for routing?

. . .

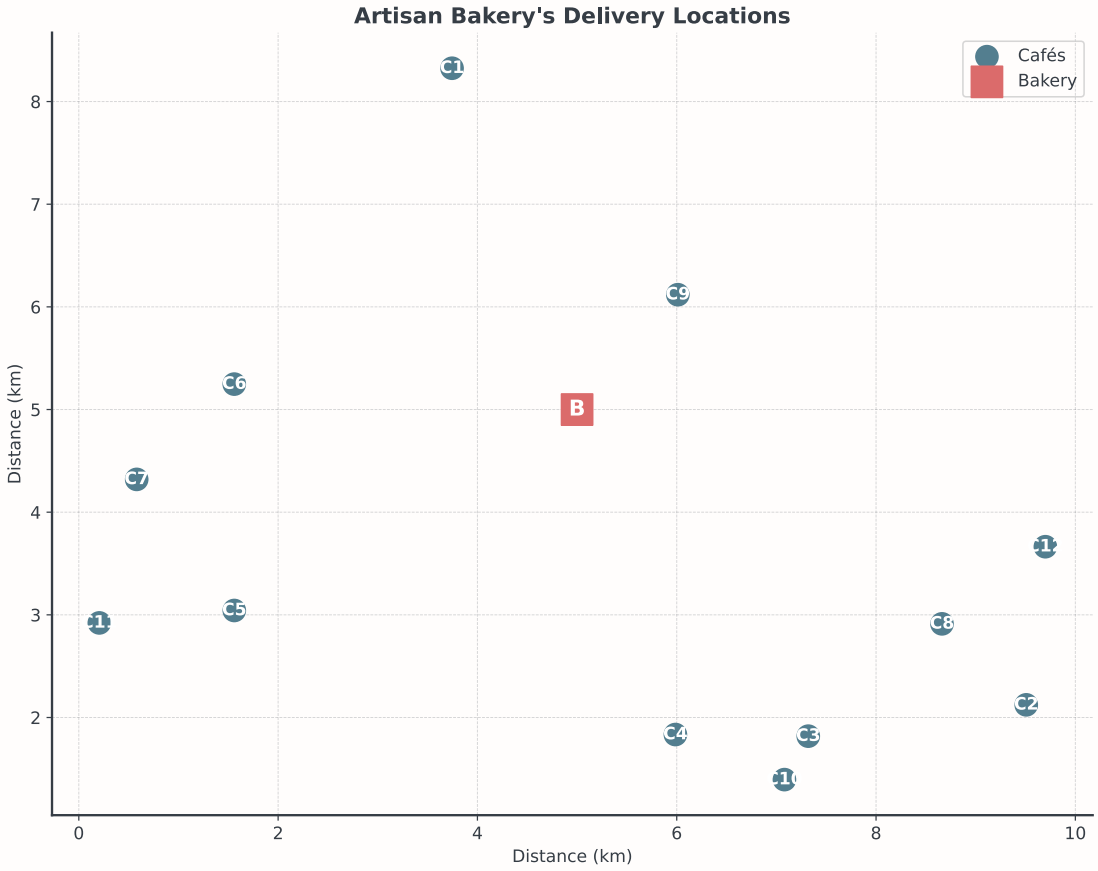
> **Note**
>
> Today's Journey: We'll start greedy (nearest neighbor), then learn how to improve solutions with local search!

The Routing Problem

The Traveling Salesman Problem

The classic optimization challenge: Visit all locations exactly once, minimize total distance.

. . .



The Complexity Explosion

Question: With 12 cafés to visit, how many possible routes exist?

. . .

Answer: 12! = 479,001,600 possible routes

. . .

Number of Stops - 5 cafés - 8 cafés - 12 cafés - 20 cafés

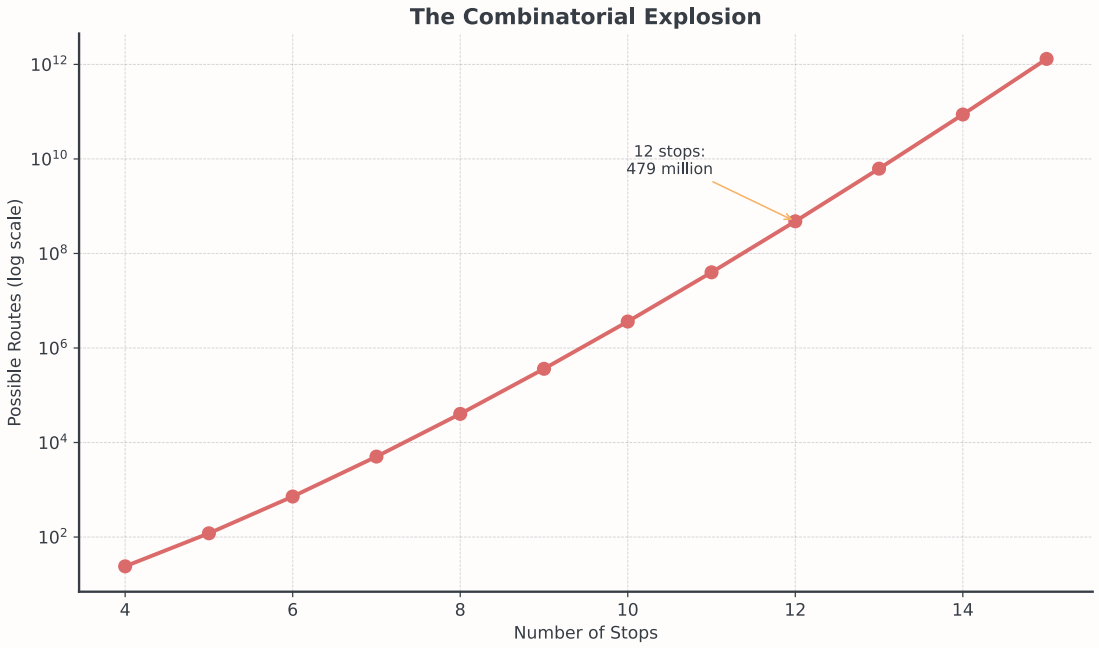
Possible Routes - 120 - 40,320 - 479 million - 2.4 quintillion

. . .

> **Warning**
>
> Testing all routes for 12 cafés would take hours even on a fast computer. For 20 cafés? 77,000 years!

Why TSP is Hard

The factorial growth makes exhaustive search impossible.



Greedy Construction

Nearest Neighbor Algorithm

Build a route by always visiting the closest unvisited location.

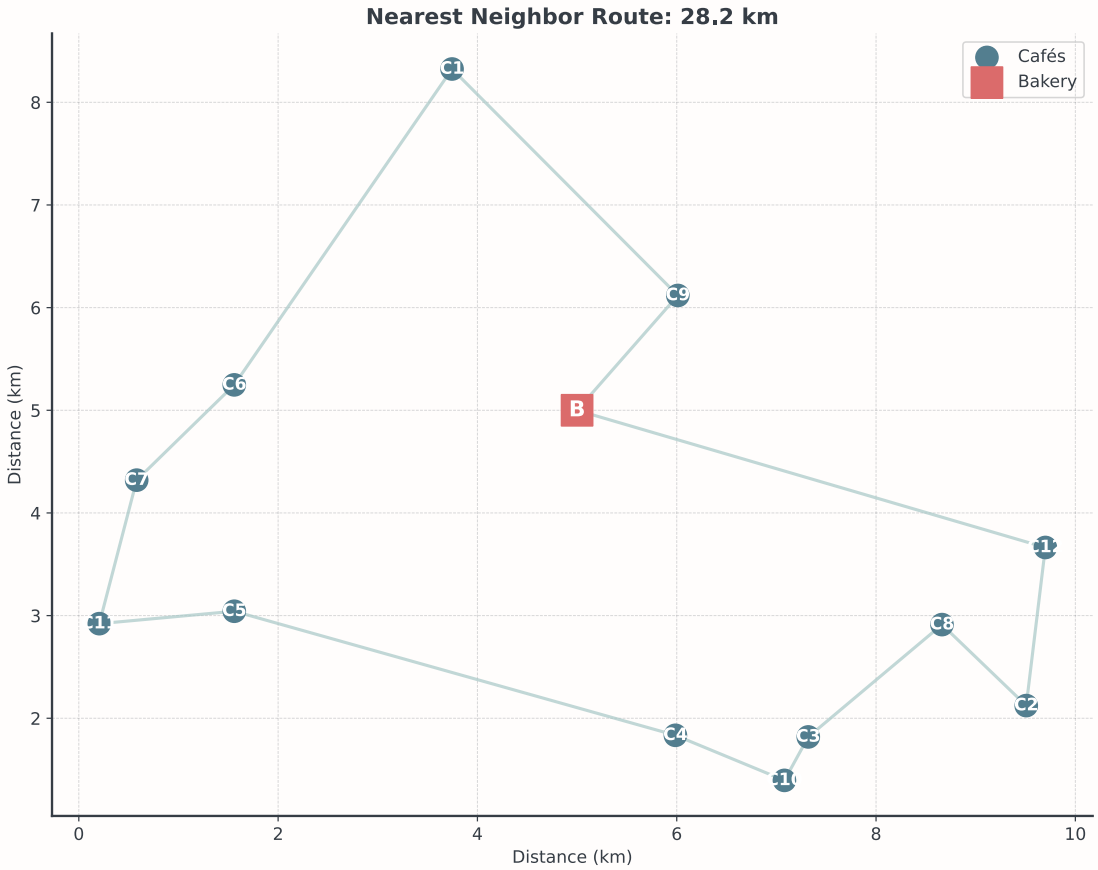
. . .

The Algorithm: 1. Start at the bakery 2. Find the nearest unvisited café 3. Go there 4. Repeat until all visited 5. Return to bakery

. . .

> **Tip**
>
> Intuition: Like picking low-hanging fruit - grab what's easiest (nearest) first!

Nearest Neighbor in Action



The Problem with Greedy

Question: Can you spot any obvious inefficiencies in this route?

. . .

Common Issues: ::: incremental - Crossing paths: Route crosses over itself - Long return: Far from bakery at the end - Myopic decisions: Can't see the "big picture" :::

. . .

> ⚠ Warning
>
> Nearest neighbor typically gives solutions 15-20% worse than optimal. Can we improve it?

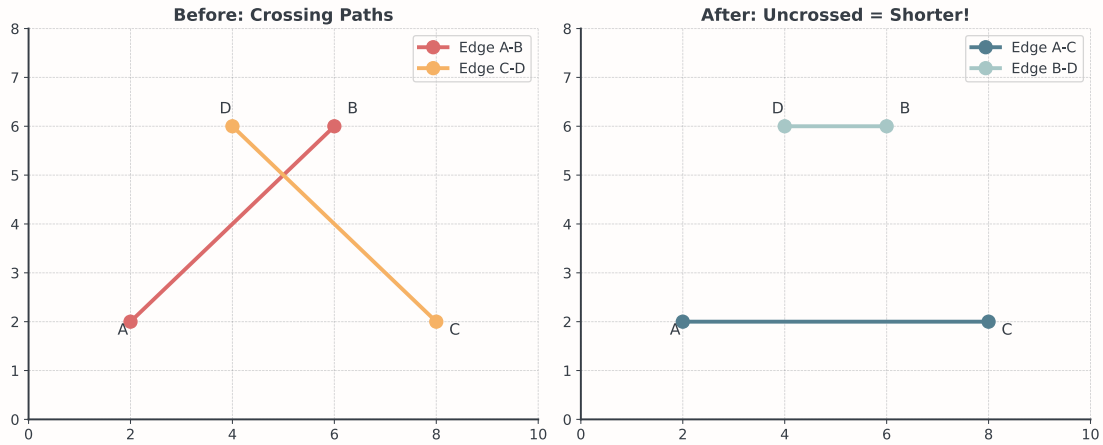
Local Search Improvements

The 2-Opt Algorithm

Systematically improve routes by removing crossing paths.

. . .

The Idea: Take two edges and swap them to uncross the route



How 2-Opt Works

The systematic improvement process:

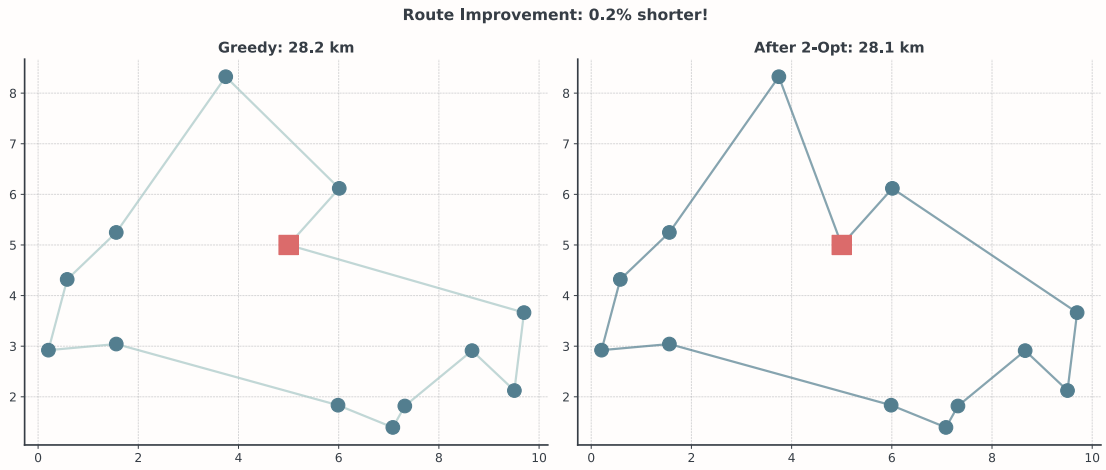
...

```
# Simplified 2-opt logic (conceptual)
for i in range(route_length):
    for j in range(i+2, route_length):
        # Try swapping edges (i,i+1) and (j,j+1)
        new_distance = calculate_with_swap(i, j)
        if new_distance < current_distance:
            perform_swap(i, j)
            improvement_found = True
```

...

> **Note**
>
> 2-opt examines all possible pairs of edges and swaps those that reduce total distance. Keep iterating until no improvement found!

2-Opt Applied to Our Route



Local Search Philosophy

Start with any solution, then iteratively improve it.

. . .

The Process: ::: incremental 1. Initial Solution: Use greedy (fast but suboptimal) 2. Define Neighborhood: What changes can we make? 3. Search: Try neighboring solutions 4. Accept: Move if better 5. Repeat: Until no improvements found :::

. . .

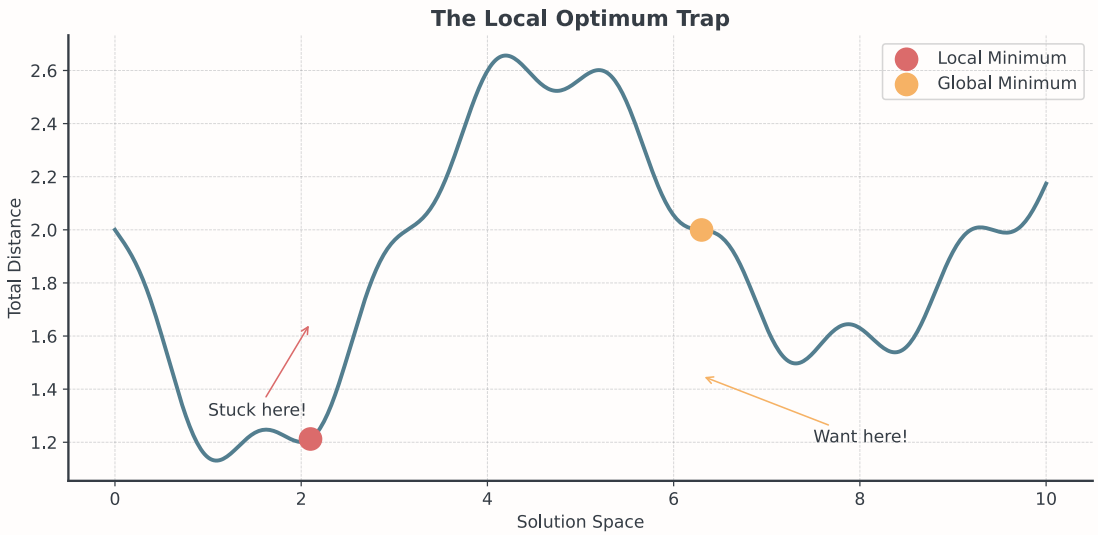
> **Tip**
>
> Local search transforms “quick and dirty” solutions into “pretty good” ones!

When Local Search Gets Stuck

Question: What happens when no single swap improves the solution?

. . .

Local Optimum: Best in the neighborhood, but not globally optimal



...

> ⚠ Warning
>
> 2-opt can only make small changes. Sometimes you need bigger moves or different approaches (next week: metaheuristics)!

Time Windows Add Complexity

Some cafés open early and need priority delivery:

...

Artisan Bakery's Constraints: - Café Europa: Opens 6:30 AM (early birds) - Sunrise Bistro: Opens 6:30 AM (breakfast rush) - Morning Glory: Opens 6:45 AM (commuter stop) - Others: Open 7:00 AM or later

...

> ! Important
>
> With time windows, we must balance distance minimization with deadline satisfaction. Pure distance optimization might arrive too late!

Mission Briefing

Your Toolkit for Today

Construction + Improvement = Better Routes

...

Construction (Greedy) - Nearest Neighbor - Cheapest Insertion - Savings Algorithm

Improvement (Local Search) - 2-opt swaps - 3-opt (advanced) - Or-opt (move sequences)

. . .

> **Note**
>
> Most commercial routing software uses this two-phase approach: Build quickly, then polish!

Next: Bean Counter Practice

In the notebook, you'll help Bean Counter optimize coffee bean deliveries to 10 franchises.

. . .

You'll implement: ::: incremental - Distance calculations between locations - Nearest neighbor construction - Route distance measurement - 2-opt improvement - Performance comparison :::

. . .

Then: Apply these skills to Artisan Bakery's 12-café challenge!

The Competition Challenge

Artisan Bakery needs your help with their morning route!

. . .

Your Mission: - Optimize delivery to 12 cafés - Handle 3 early time windows - Minimize total distance - Beat the current 3-hour route

. . .

> **Tip**
>
> Hint: Start with nearest neighbor, but don't forget about those time windows! Sometimes a slightly longer route that meets deadlines is better than the shortest route that arrives late.

Literature

Resources

Essential Reading: - Applegate et al. (2011): The Traveling Salesman Problem - The definitive TSP reference - Laporte (1992): The Vehicle Routing Problem - Overview of routing algorithms - Lin & Kernighan (1973): Classic paper on k-opt improvements

Python Libraries: - `scipy.spatial.distance` - Fast distance calculations - `networkx` - Graph algorithms including TSP approximations - `ortools` - Google's optimization tools with routing

> **Summary**
>
> - TSP is computationally hard (factorial growth)
> - Greedy construction gives fast initial solutions
> - Local search (2-opt) improves solutions iteratively
> - Real constraints (time windows) add complexity
> - Two-phase approach: Build then improve!

Bibliography